SOFTWARE & INTERNET QUALITY WEEK EUROPE

Brussels, BELGIUM • November 20-24, 2000

INITIATIVES FOR THE FUTURE

The New Century has begun! Internet Quality and Software Quality worries are pressing issues!

QWE2000 focuses on quality technology with a careful, honest look at current methods and cautious assessments of future trends.

QWE2000: The 4th year of serving the SQA community with state-of-the-art information!

Organized by

Software Research Institute

In cooperation with

ASQ Software Division

ESI European Software Institute

SAI

TestNet

European ESSI Systems & Software Institute

Industry Sponsors

CMG Information Technology

eValid

gitek Interaction through software

ps_testware Software Testing Services

4th INTERNATIONAL SOFTWARE QUALITY WEEK EUROPE (QWE2000)

Brussels, Belgium
20-24 November 2000

TECHNICAL PROGRAM

Theme: *Initiatives for the Future*

IMPORTANT NOTE: This program, speaker biographies, and speaker abstracts are based on the best available information available but may be subject to change. Updated 05 August 2000.

QUICK ACCESS TO THE FIVE DAY PROGRAM AT QWE2000

The QWE2000 program is linked to speaker biographies and presentation abstracts.
Click on a ***Speaker*** to read a Biography.
Click on a ***Title*** to read the Presentation Abstract.

REGISTER FOR QWE2000

Monday, 20 November 2000
TUTORIAL DAY #1

	Tutorial A1	Tutorial B1	Tutorial C1	Tutorial D1
T1 8:30 - 10:00 Coffee	Dr. Gualtiero Bazzana (ONION, S.P.A.) *Web Testing Master Class*	Mr. Ruud Teunissen and Rob Baarda (Gitek) *Risk Based Test Effort Estimation with Test Point Analysis*	Dr. Alan Cameron Wills (TriReme International Ltd) *Component Architecture and Business Models*	Mike Russell (Insight Consulting Ltd., Ireland in association with Systeme Evolutif, UK) *ISEB Software Testing Foundation Certificate*

10:30 - 12:00	Part I of II			Part I of V
12:00 - 13:30	TUTORIAL DAY LUNCH AND NETWORKING			
13:30 - 15:00 Refreshments 15:30 - 17:00 (evening) 18:00-21:00	**Tutorial A2** Dr. Gualtiero Bazzana (ONION, S.P.A.) *Web Testing Master Class* Part II of II	**Tutorial B2** Ms. Alice Lee and Dr. Eric Wong (NASA Johnson Space Center) *A Quantitative Risk Assessment Model For Software Quality, Testing and Safety*	**Tutorial C2** Dr. Hans-Ludwig Hausen (GMD) *On A Standards Based Quality Framework for Web Portals*	**Tutorial D2** Mike Russell (Insight Consulting Ltd., Ireland in association with Systeme Evolutif, UK) *ISEB Software Testing Foundation Certificate* Part II of V ********** Part III of V: (evening) 18:00-21:00

[BACK TO TOP]

Tuesday, 21 November 2000 TUTORIAL DAY #2				
T2 8:30 - 10:00 Coffee 10:30 - 12:00	**Tutorial E1** Mr. Tom Gilb (Result planning Limited) *Requirements Engineering for SW Developers and Testers*	**Tutorial F1** Mr. Robert A. Sabourin (AimBug.Com) *The Effective SQA Manager Getting Things Done*	**Tutorial G1** Mr. Adrian Cowderoy (ProfessionalSpirit Limited) *Cool Q-Quality Improvement For Multi-Disciplinary Tasks In Website Development*	**Tutorial H1** Mike Russell (Insight Consulting Ltd., Ireland in association with Systeme Evolutif) *ISEB Software Testing Foundation Certificate* Part IV of V
12:00 - 13:30	TUTORIAL DAY LUNCH AND NETWORKING			
13:30 - 15:00 Refreshments 15:30 - 17:00	**Tutorial E2** Mr. Tom Gilb (Result Planning Limited) *Specification Quality Control (SQC): A Practical Inspection Workshop on Requirements Specification*	**Tutorial F2** Mr. Tom Drake (Integrated Computer Concepts, Inc.) *The Quality Challenge for Network-Based Software Systems*	**Tutorial G2** Mr. Tobias Mayer and E. Miller (eValid, Inc.) *WebSite Testing*	**Tutorial H2** Mike Russell (Insight Consulting Ltd., Ireland in association with Systeme Evolutif) *ISEB Software Testing Foundation Certificate* Part V of V EXAM: Wednesday

	Part II of II			Morning 8:30 - 10:00 AM
17:00 - 18:00	Welcome Reception			

[BACK TO TOP]

Wednesday, 22 November 2000
CONFERENCE DAY #1

Exhibition: 10:00 AM to 18:00 PM

K1 8:30	**PLENARY SESSIONS** **Conference Introduction / Session Introduction:** Edward Miller (Conference Chairman) (Software Research, Inc.) **Keynote K1-1:** Mr. Tom Gilb (Results Planning, Norway) *The Ten Most Powerful Principles for Quality in Software Organizations* **Keynote K1-2:** Mr. Jens Pas (I2B, Belgium) *Test Out-Sourcing: From Necessary Evil to E-Competitive Advantage* **Keynote K1-3:** Dr. Philippe Aigrain (Head of "Software Technologies" Sector, EC, Brussels, Belgium) *A Wider Look at Software Quality*				
10:00	REFRESHMENTS IN EXHIBIT HALL				
1 10:30	**Technology Track** *Track Theme* **Paper 1T** Ms. Miriam Bromnick (Ovum ltd,UK) *Automated Software Testing: A New Breed of Tools*	**Applications Track** *Track Theme* **Paper 1A** Mr. Jean Hartmann, Mr. Claudio Imoberdorf (Siemens Corporate Research, USA) *Functional Testing Of Distributed Component-Based Software*	**Internet Track** *Track Theme* **Paper 1I** Dr. Lingzi Jin (FamilyGenetix Ltd., UK) *Introducing Quality Assurance Into Website Development: A Case Study For Website Quality Control In A Small Company Environment*	**Management Track** *Track Theme* **Paper 1M** Mr. Ton Dekkers (IQUIP Informatica B.V.) *Quality Tailor-Made (QTM)*	**Vendor Technical Presentations** **VT 1** Mr. Robin Bortz RadView *WebLoad Integrity Testing for e-applications*
	REFRESHMENTS IN EXHIBIT HALL				

	EXHIBITS OPEN				
	Technology Track *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
2 11:30	**Paper 2T** Mr. Tom Hazdra & Lubos Kral (CertiCon, Czech Republic) *Enhancing the Integration Testing of Component-Based Software*	**Paper 2A** Mr. Olaf Mueller & Mr. Axel Podschwadek (Siemens, Germany) *A Step-to-Step Guide to Incremental Testing: Managing Feature Interaction for Communication Devices*	**Paper 2I** Ms. Nicole Levy (Laboratore PRISM, France) *Quality Characteristics to Select an Architecture for Real-Time Internet Applications*	**Paper 2M** Mr. Kie Liang Tan (CMG TestFrame Finance, Netherlands) *How To Manage Outsourcing Of Test Activities*	**VT 2** Lisa Hoven Rational *Successful Testing Through Requirements*
12:30	CONFERENCE LUNCH AND NETWORKING IN EXHIBIT HALL				
	Technology Track *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
3 14:00	**Paper 3T** Dr. Antonia Bertolino, F. Basanieri (CNR-IEI, Italy) *A Practical Approach to UML-based Derivation of Integration Tests*	**Paper 3A** Mr. Rob Hendriks & Mr. Robert van Vonderen & Mr. Erik van Veenendaal (Improve Quality Services, Netherlands) *Measuring Software Product Quality During Testing*	**Paper 3I** Mr. Massimiliano Spolverini (Etnoteam - Consulting Division, Italy) *Measuring And Improving The Quality Of Web Site Applications*	**Paper 3M** Mr. Kees Hopman (IQUIP Informatica BV, Netherlands) *How to Implement New Technologies? Four Proven Cornerstones for Effective Improvements*	**VT 3** David Walker Technology Builders, Inc. *Effective Requirements Management Using Caliber-RMTo Be Announced*
	REFRESHMENTS IN EXHIBIT HALL				
	Technology Track *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
4	**Paper 4T** Ms. Lisa Crispin (iFactor-e) *The Need for Speed: Automating Functional*	**Paper 4A** Mr. Steve Littlejohn (SIM Group Limited., UK) *Test Environment Management -- A Forgotten Basic*	**Paper 4I** Mr. Adrian Cowderoy (ProfessionalSpirit) *Complex WebSites Cost More to Maintain - Measure the Complexity of*	**Paper 4M** Mr. Andreas Birk & Wolfgang Mueller (Fraunhofer Institute, Germany) *Systematic Improvement*	**VT 4** Hans Buwalda CMG *TestFrame: Getting Testing and Test Automation*

15:00	*Testing in an eXtreme Programming Environment*		*Content*	*Management: A Method For Defining And Controlling Customized Improvement Programs*	*Under Control*
	REFRESHMENTS IN EXHIBIT HALL				
5 16:00	**Technology Track** *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
	Paper 5T Mr. Ruud Teunissen (Gitek) *Improving Developer's Tests*	**Paper 5A** Dr. Erik P. vanVeenendaal (Improve Quality Services BV, Netherlands) *GQM Based Inspection*	**Paper 5I** Mr. Rakesh Agarwal, Bhaskar Ghosh, Santanu Banerjee & Soumyendu Pal (Infosys Technologies Ltd, India) *Challenges And Experiences In Establishing WebSite Quality*	**Paper 5M** Mr. Oliver Niese, Tiziana Margaria, Markus Nagelmann, Bernhard Steffen, Georg Brune & Hans-Dieter Ide (META Frame Technologies GmbH) *An Open Environment for Automated Integration Testing*	**VT 5** Mr. Alexander Malshakov & Mr. George St. Clare (Amphora Quality Technologies) *Optimizing Iteration Testing*
17:00 - 18:00	COCKTAIL PARTY IN EXHIBIT HALL				

[BACK TO TOP]

Thursday, 23 November 2000
CONFERENCE DAY #2

Exhibition: 10:00 AM to 18:00 PM

K2 8:30	**PLENARY SESSIONS**
	Session Introduction: Edward Miller (Software Research, USA)
	Keynote K2-1: Ms. Lisa Crispin (iFactor-e) *Stranger in a Strange Land: Bringing QA to a Web Startup*
	Keynote K2-2: Mr. Hans Buwalda (CMG Finance)

	Soap Opera Testing				
	QWE2K Awards Announcement of the QWE2000 Best Paper Award				
10:00	REFRESHMENTS IN EXHIBIT HALL *EXHIBITS OPEN*				
6 10:30	**Technology Track** *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
	Paper 6T Mr. Francesco Piazza (Alenia Aerospazio, Italy) *A Requirements Trace Application*	**Paper 6A** Mr. Jacobus DuPreez & Lee D. Smith (ARM Ltd.) *SPI: A Real-World Experience*	**Paper 6I** Mr. Olivier Denoo (ps_testware,Belgium) *Usability: A web Review*	**Paper 6M** Mr. Karl Lebsanft & Mr. Thomas Mehner (Siemens AG, Germany) *CMM in Turbulent Times - Is CMM a Contradiction to Innovation?*	**VT 6** Panos Ntourntoufis UPSPRING Software, Inc. *An Automated Approach to Software Defect Prevension*
	REFRESHMENTS IN EXHIBIT HALL				
7 11:30	**Technology Track** *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
	Paper 7T Mr. William E. Lewis (Technology Builders, Inc., Canada) *Requirements-Based Testing: An Overview (Process and Techniques for Successful Development Efforts*	**Paper 7A** Ms. Jill Pritchet & Mr. Ian Lawthers (Centre for Software Engineering) *Software Process Improvement for Small Organizations using the "SPIRE" Approach*	**Paper 7I** Mr. Fernando T. Itakura, Ms. Silvia R. Verfilio (Crosskeys Systems Corporation,Brazil) *Automatic Support For Usability Evaluation of A Web Application*	**Paper 7M** Mr. Luis Filipe D. Machado, Ms. Kathia M. de Oliveira & Ms. Ana Regina C. Rocha (Federal University Of Rio de Janeiro, Brazil) *Using Standards And Maturity Models For The Software Process Definition*	**VT 7** Mr. Matthew Brady McCabe & Associates UK Ltd *How to Test Better - Not Test More*
12:30	CONFERENCE LUNCH AND NETWORKING				
	Technology Track *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
	Paper 8T	**Paper 8A**	**Paper 8I**	**Paper 8M**	**VT 8**

8 14:00	Mr. Tobias Mayer (eValid, Inc.) *Browser-Based WebSite Testing Technology*	Mr. Gunthard Anderer (CMG ORGA - Team GmgH, Germany) *Testing E-Commerce Systems - Requirements And Solutions*	Mr. Eric Messin (Vality Technology, USA) *Ensuring Data Quality for E-Commerce Systems*	Mr. Martin S. Feather, Mr. Tim Kurtz (NASA Glenn Research Center, USA) *Putting It All Together: Software Planning, Estimating And Assessment For A Successful Project*	Ruud Teunissen Gitek *TWeb: Testing e-Business Applications*
	REFRESHMENTS IN EXHIBIT HALL				
	Technology Track *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
9 15:00	**Paper 9T** Dr. Rainer Stetter (Software Factory & ITQ Gmbh, Germany) *Test Strategies for Embedded Systems*	**Paper 9A** Dasha Klyachko (Allied Testing, UK) *Specifics of E-Testing: Difference Between Traditional and On-Line Software Development and Its Effect on Testing*	**Paper 9I** Mr. Adrian Cowderoy (ProfessionalSpirit Limited,UK) *Ensuring Data Quality Is Just The Start -- The Real Battle Is Commercial Quality*	**Paper 9M** Dr. Esther Pearson (Genuity Corporation, USA) *Website Operational Acceptance Testing; Process Assessment and Improvement*	**VT 9** Peter Sterck [Belgium] (ps_testware) *The Challenge of e-business Testing*
	REFRESHMENTS IN EXHIBIT HALL				
	Technology Track *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
10 16:00	**Paper 10T** Dr. Ray Paul & Dr. Wei-Tek Tsai (USA) *Assurance-Based Testing: A New Quality Assurance Technique*	**Paper 10A** Mr. Bob Bartlett (SIM Group Limited., UK) *A Practical Approach to Testing your eCommerce Web Server*	**Paper 10I** Mr. Robert L. Probert, Wujun Li, Mr. Paul Sims (School of Information Technology and Engineering, Canada) *A Risk Directed E-Commerce Test Strategy*	**Paper 10M** Mr. William E. Lewis (Technology Builders, Inc., Canada) *A Continuous Quality Improvement Testing Methodology*	**VT 10** Edward Miller (eValid, Inc.) *The Argument for Client-Side Testing*

17:00 - 18:00	Special Evening Event (To Be Announced)

[BACK TO TOP]

Friday, 24 November 2000
CONFERENCE DAY #3

	Technology Track *Track Theme*	Applications Track *Track Theme*	Internet Track *Track Theme*	Management Track *Track Theme*	Vendor Technical Presentations
11 8:30	**Paper 11T** Mr. Richard Kasperowski & Mr. Spencer Marks (Altisimo Computing, USA) *Building Better Java Applications*	**Paper 11A** Dr. Nigel Bevan (Serco Usability Services, UK), Mr. Itzhak Bogomolni (Israel Aircraft Industries, Israel) *Incorporating User Quality Requirements In The Software Development Process*	**Paper 11I** Mr. Bob Bartlett (SIM Group Limited., UK) *Experience Testing E-Commerce Systems*	**Paper 11M** Mr. Vassilios Sylaidis, Mr. Dimitrios Stasinos, Mr. Theodoros [Greece] (INTRACOM S.A.) *Software Development Process Improvement For Telecommunications Applications By Applying Gilb's Inspection Methodology*	**VT 11** Bruno Bouyssounouse (PolySpace) *Detect All Run-time Error Without Test-Beds*
	REFRESHMENTS				
	Technology Track *Track Theme*	**Applications Track** *Track Theme*	**Internet Track** *Track Theme*	**Management Track** *Track Theme*	**Vendor Technical Presentations**
12 9:30	**Paper 12T** Mr. Sanjay DasGupta & Indrajit Sanyal (Usha Communications Technology, INDIA) *A Java-XML Integration for Automated Testing*	**Paper 12A** Mr. Adam Kolawa (ParaSoft, USA) *Testing Dynamic Web Sites*	**Paper 12I** Mr. Steven D. Porter (Practical Consulting Group, USA) *From Web Site To Web App: Ensuring Quality In A Complex Enviroment*	**Paper 12M** Ms. Tuija Lamsa (University of Oulu, Finland) *Using Knowledge Management in the Quality Improvement of the Development Processes*	**VT 12** Mr. Bob Bartlett SIM Group *The dream comes true - Scriptless Automated Testing*
10:30	REFRESHMENTS				
	PLENARY SESSIONS				
	Keynote K3-1: Rik Daems Minister of Telecommunications				

K3 11:00	Government of Belgium *Belgium's Five-Star Plan to Develop The Information Society*
	Keynote K3-2: Mr. Tom Drake (ICCI) *The Future of Software Quality - Our Brave New World - Are We Ready?*
	QWE2K Awards Announcement of the QWE2000 Best Presentation Award
	Conference Conclusion: Edward Miller (Software Research, USA)

[BACK TO TOP]

Extra Presentations

EP	Wednesday	Thursday	Friday
	Mr. Leif Balter[Sweden] (Cap Gemini Ernst & Young) *Create Your Own Testtool*	Mr. Olivier Denoo [Belgium] (ps_testware) *Assuring Your E-commerce Revenue*	Mr. Richard Kasperowski (Altisimo Computing) *Opportunistic Software Quality*

QWE2000 Tutorial A1 - A2

Dr. Gualtierio Bazzana
(ONION, S.P.A)

"Web Testing Master Class"

Key Points

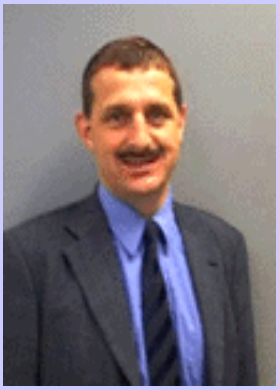
- To gain industry recognition for testing as an essential and professional software engineering specialisation.
- Through the BCS Professional Development Scheme and the Industry Structure Model, provision of a standard framework for the development of testers' careers.
- To enable professionally qualified testers to be recognised by employers, customers and peers, and to raise the profile of testers.
- To promote consistent and good testing practice within all software engineering disciplines.
- To identify testing topics that are relevant and of value to industry.
- To enable software suppliers to hire certified testers and thereby gain commercial advantage over their competitors by advertising their tester recruitment policy.
- To provide an opportunity for testers or those with an interest in testing to acquire an industry recognised qualification in the subject.

Presentation Abstract

First of all, peculiarities of Web-based applications will be presented from a technical point of view, explaining their effects on testing practices. Moreover, the tutorial will deal with testing management aspects which are fundamentally affected by the nature of Web applications, including: RAD, regression issues, Testing solution will then be presented, both for static aspects (related to HTML, pictures, XML) and dynamic aspects (ASP, CGI, Proxies, Cookies, etc.).Room will be devoted also to commercial tools available in order to give the audience an overview of the existing technologies, highlighting also experience reports from their introduction, including ROI analysis. Special emphasis will then be given to the Web Accessibility Initiative (WAI) guidelines that have been issued by W3C and can significantly help testing of Web-based applications. Last but not least, the tutorial will touch the issues raised by integration testing between ERP and Web and in the validation of E-business solutions. Case studies will cover: testing of e-commerce sites, testing of commercial Internet Web sites, testing of Intranet sites, testing of home banking/ trading on-line applications

About the Speaker

Born in 1966, at university, he graduated with 110/110 and honour in Information Science at the University of Milan, in February 1989. His PhD won the special AICA award for topics related to quality in Information Technology. After working as software developer in a telecommunication company and as consultant/ manager in a consulting company he set-up ONION. His activities cover two areas of interest: consulting - projects in software engineering for various industrial companies and research in the field of software quality and networking (especially in the Internet/ Intranet domain). As far as consulting/ projects in industry are concerned, he matured and exploited know-how in conducting various medium sized and large projects for several companies in various application domains (telecommunications, data processing, MIS, process control, etc.), covering topics like: Internet services, Intranet applications, Supply Chain management, etc. Moreover he has matured significant experiences in the ERP domain, notably with SAP R/3. He has matured significant technical experiences especially in the telecommunications domain (notably: switching systems and GSM mobile radio systems) in CIM and in networking, including Internet/ Intranet/ Extranet services and solutions. He has also dealt with sw development and testing, testing methods and tools, quality planning, test planning, reliability analysis, software product evaluation, process assessment and improvement, definition of quality systems in accordance to ISO 9001 and 9000/3, reviews and inspections, FDA computer system validation and so forth His research activity spanned in various fields of software engineering, ranging from Petri Nets to development methodologies, functional and structural test coverage, metrics and related tools, CAST, reliability evaluation, software development process evaluation and improvement, management by metrics, software product quality evaluation, security technology transfer and total quality management techniques. Moreover, he has co-ordinated several European Research Projects. He has published a book: ("Software Metrics for Product Assessment", McGraw Hill, London, 1995, International Software Quality Assurance Series, ISBN 0-07-707923-X), contributed to 4 other books (in the last one he has been author of four chapters dedicated to Software Process Improvement; it has been published by IEEE Software) and published over 50 papers at international conferences on topics related to software quality and software testing



QWE2000 Tutorial B1

Mr. Ruud Teunissen and Rob Baarda (Gitek)

"Risk Based Test Effort Estimation with Test Point Analysis"

Key Points

- Well defined steps from business risks to test coverage
- Early and stronger test involvement of all parties concerned
- Useful for all tests

Presentation Abstract

Testing of an information system should be based on the business risks for the organisation in using that information system. In practice, the test manager often takes the steps to go from risks to test coverage in an intuitive way. In this presentation, the steps to define a testing strategy are made explicit. This gives all parties involved better insight and provides a sound basis for negotiating testing depth.

A good risk assessment is a part of these steps. Very important is that this explicit way of looking at risks clearly shows that a test manager or tester can't do this alone. The involvement of users and managers of the client organisation and of project people like the developers, testers, QA'ers and project manager is necessary. Discussing risks and testing in the above way proves in practice to be real eye-openers for all parties concerned. This also enables negotiating about testing depth by letting the customer choose what should be tested how thoroughly.

The stepwise defining of the test strategy can be used for any test level and also for an overall strategy, including and co-ordinating all test levels and even inspections.

About the Speaker

Since 1989 Ruud Teunissen is employed in the testing world. He has been involved in a large number of ICT projects and has performed several functions within the testing organisation: tester, test specialist, test advisor, test manager, etc. Based on his experience, Ruud participated in the development of the structured testing methodology TMap® and is co-author of several books on structured testing. The

last years Ruud is involved in implementing structured testing within organisations on the Belgian and Dutch market. At this moment Ruud is working in Belgium for Gitek n.v. as Manager Testen. Ruud is frequently speaking in Benelux and Great Britain.

Rob Baarda is an information systems professional for more than 20 years, following the path from programmer to consultant. Since 1986 he specialised in the field of testing. Starting with developing automated tests, Rob moved after a few years to the methodology of testing. He is now part-time researching various test subjects in the R&D department of IQUIP, besides working as an international test consultant and teaching TMap½ and TPI½. He also presented in QWE'99 about Risk Based Test Strategy.

QWE2000 Tutorial C1

Dr. Alan Cameron Wills (TriReme International Ltd)

"Component Architecture and Business Models"

Key Points

- Flexible systems are built from pluggable components
- Pluggable components need interoperable architecture
- Interoperability requires unambiguous business models

Presentation Abstract

The pace of change in business requires a very rapid software production cycle. The only way to meet rapidly changing requirements is to build software from an evolving kit of predefined components. The software architect must design an extensible family of products, not just a single application.

How do we specify and build such architectures? How do we assure reliability?

This session will provide a clear definition of component architecture as a framework for interoperable components. We will see how to use UML to define the interfaces between them, based on models of the business; and how to derive from these models unambiguous conformance tests for the components, as well as definitions of interoperation protocols (for example in XML).

The material is based on the Catalysis approach to CBD.

About the Speaker

Alan Cameron Wills has been a consultant in software development methods since 1990, working with clients in a wide variety of application areas, on both sides of the Atlantic. He is joint developer and author of the Catalysis approach to component based development. Dr Wills is Technical Director of Trireme International Ltd, a UK-based consultancy.

QWE2000 Tutorial D1

Mike Russell
(Insight Consulting Ltd., Ireland in association with Systeme Evolutif, UK)

"ISEB Software Testing Foundation Certificate"

The Information Systems Examination Board (ISEB), a division of British Computer Society, has accredited the standard full course for delegates of Quality Week Europe, with some experience in testing, who wish to take the examination leading to the Foundation Certificate in Software Testing. The course will take 18 hours, including a 'closed book' exam; which consists of forty multiple-choice questions.

The exam will be offered during the conference, supervised by an ISEB invigilator. The results and certificates will be presented at a special announcement during QWE2000.

Objectives of the Qualifications

Through the Software Testing qualifications offered by ISEB, the Subject Board has the following objectives:

- To gain industry recognition for testing as an essential and professional software engineering specialization.
- Through the BCS Professional Development Scheme and the Industry Structure Model, provision of a standard framework for the development of testers' careers.
- To enable professionally qualified testers to be recognized by employers, customers and peers, and to raise the profile of testers.
- To promote consistent and good testing practice within all software engineering disciplines.
- To identify testing topics that are relevant and of value to industry.
- To enable software suppliers to hire certified testers and thereby gain commercial advantage over their competitors by advertising their tester recruitment policy.
- To provide an opportunity for testers or those with an interest in testing to acquire an industry recognized qualification in the subject.

Presentation Abstract

Syllabus Topics for the Foundation Certificate in Software Testing:

- Principles of Testing
- Testing thoughout the lifecycle
- Dynamic Testing Techniques
- Static Testing
- Test Management
- Tool Support for Testing (CAST).

Fees

The fee for sitting the examination is currently 200 Euros - 17.5% VAT included, payable to ISEB., through Software Research Institute. (see registration form)

About the Speaker

Mike Russell (B.Sc., M.Sc. in Computer Science) has over 12 years experience in the software industry, working for companies such as Digital, Motorola and Nellcor Puritan Bennett. His expertise covers methods and techniques for both software development and independent system level testing that includes significant hands-on experience of test management, test specification/implementation and tools. He has designed and delivered numerous test-related training courses for organizations and performed numerous project assignments including the introduction of structured testing methodologies for the V-model and OO development lifecycles. In addition, Mike is also involved in the areas of Software Project Management and Quality Assurance.

QWE2000 Tutorial B2

Ms. Alice Lee and Dr. Eric Wong
(NASA Johnson Space Center and Telcordia Technologies)

"A Quantitative Risk Assessment Model For Software Quality, Testing and Safety"

Key Points

- Quantitative Risk Management for software
- Test Efficiency and Improvement
- Software Quality and Maturity Level Improvement

Presentation Abstract

The risk management model was created and validated by the instructor for NASA. The model was created to meet the challenge of the prevailing schedule/cost constraints without sacrificing product quality for highly safety-critical space flight software. The model, which has been applied to critical NASA flight software and ground test facilities, presents a quantitative approach to assessing the software's quality, test efficiency, and functional criticality. It can be used throughout the software development life cycle phases.

The model quantitatively measures the software down to the module level, and derives a risk index to present the module's risk level. The risk index provides a quick reference for project managers to understand the status of the software quality, so as to determine the budget and schedule more accurately prior to delivery. The risk index also provides the developers and testers with insight into where the problems are, and what causes the problems. With this approach, unnecessary testing was eliminated and test planning was more effective. The risk index along with the measurements of the composite elements served as a useful tool for managing schedule and budget risk and allowed the designers and testers to focus on improvement areas and mitigating risks. In addition, the application of the model can promote the Capability Maturity Level of the software to levels 4 and 5.

The assessment work is performed with automated tools. Part I of this seminar

describes the concept, approach, and methodology of the risk model. The quality measurement tool will also be introduced in-depth. Part II of the seminar will discuss in detail a more efficient approach for software testing and maintenance.

About the Speaker

Alice Lee is currently the Chief Technologist for SR&QA (Safety Reliability and Quality Assurance) Office at NASA Johnson Space Center. She is also Assistant Division Chief for the Technology Division of SR&QA. She attended Purdue University, USA, for undergraduate studies in Computer Science. Received a Master's Degree in Computer Science from Rice University, USA, in 1986. She was also selected to attend MIT in 1994 under a NASA fellowship for Advanced Engineering Studies. She created this quantitative risk model for NASA.

Eric Wong is currently a Research Scientist at Telcordia Technologies (formerly known as Bellcore). He is one of the principal investigators at Telcordia responsible for developing a set of metrics for evaluating the overall quality of highly complex telecommunication systems and to identify their fault-prone software modules. He received his B.S. in Computer Science from Eastern Michigan University, and his M.S. and Ph.D. in Computer Science from Purdue University.

QWE2000 Tutorial C2

Dr. Hans-Ludwig Hausen (GMD)

"On A Standards Based Quality Framework for Web Portals"

Key Points

- Are web portals so peculiar that the advantages of defining a quality model for web portal overcomes the disadvantage of having standard proliferation?
- How can the quality characteristics/sub-characteristics defined and suggested by such attributes as given by, for example, ISO 9126 be covered by the expected properties of a web portal
- What are possible quality profiles (i. e., lists of expected relative "values" of characteristics/sub-characteristics) for web portals according to a model conform to a quality standard, e.g. ISO 9126, ISO12119, ISO14598, QoS?

Presentation Abstract

General, abstract definitions of quality have often been attempted. The concept may be discussed for long and may give rise to different, subjective formulations, each of them acceptable on the basis of some points of view. What is interesting here, is that precisely defined models can be proposed for quality. A quality model is a very general concept: it can be conceived as a structured set of properties, or characteristics, that can be established for an object to declare it a quality object. The model can be used as a guideline for developing an object or a reference to evaluate properties of a software system.

For software systems, various quality models have been explicitly or implicitly proposed (by Bogen, Deutsch, Azuma, Rae, et.al.). All of them address a methodology to define a hierarchic structure of characteristics and a way to relate these characteristics to some technical aspects of the system. In literature, there is little evidence of showing whether or not the characteristics depend on the application domain the system belongs to. This point is even purposely avoided, probably due to searching for a model as general as possible. In fact, these and other efforts have led to define and approve a very popular standard for software system quality, like the ISO/IEC 9126 resp. ISO14598 or the QoS frameworks and guidelines.

The model referred by this standard defines quality as a set of six characteristics (Functionality, Reliability, Usability, Maintainability, Efficiency, Portability) and proposes a further decomposition of each characteristic into a set of

sub-characteristic, wisely not included in the standard. Among the great variety of software products, web portal have widely grown both in the number of media and technologies involved and in the target which they are addressed to.

About the Speaker

Speaker Bio to be added later

QWE2000 Tutorial E1

Mr. Tom Gilb
(Result Planning Limited)

Requirements Engineering for SW Developers and Testers

Key Points

- Use a well structured defined requirements language to plan your testing
- Use similar good requirements thinking to plan you test organization
- Insist that colleagues and customers use similar good requirements practice before handing specifications to you to work with
  - Avoid garbage in, have entry and exit controls
  - base controls on numeric Inspection defect levels

Presentation Abstract

System Requirements Engineering Course will teach an innovative fresh approach to system requirements. It is distinguished from all previous requirements approaches by its level of quantification of critical system requirements. Its all based on a practical, defined 'language for specification which produces rigor and clarity to any requirements specification process. All cases of system design requirements are encompassed, including quality requirements, resource requirements and design constraints.

About the Speaker

Tom Gilb is an independent consultant, teacher and author. He works mainly in UK, Europe and North America. He is resident in Norway.

Tom coined the term 'Software Metrics' with the publication of his book of the same name in 1976 (European edition) and 1977 (USA edition). This work is the acknowledged (by R. Radice and W. Humphrey) as inspiration for much of the Software Engineering Institute's Capability Maturity Model Level 4 (SEI CMM Level 4). His other books include Principles of Software Engineering Management (1988, now in 13th printing) and Software Inspection (1993 with Dorothy Graham). His main professional interest is the development of powerful Systems Engineering methods (covering Requirements, Design, Quality Control and Project Management)

QWE2000 Tutorial F1

Mr. Robert A. Sabourin (Purkinje Inc.)

"The Effective SQA Manager - Getting Things Done"

Key Points

- SQA Management
- Effective Process
- Process improvement

Presentation Abstract

This interactive tutorial walks you through several "down to earth" practical aspects of running an SQA team.

The tutorial is presented in parable form. In this tutorial the audience will experience the real life problems encountered by a NOGO.COMs neophyte SQA Manager "Fred". "Fred" must turn around an enthusiastic but severely under staffed and under budget team of SQA professionals working in a chaotic development environment into a productive effective team! "Fred" is under the gun - he has to get things done!

About the Speaker

Robert Sabourin has been involved in all aspects of development, testing and management of software engineering projects. Robert graduated from McGill University in 1982. Since writing his first program in 1972, Robert has become an accomplished software engineering management expert. He is presently the President of AmiBug.Com, Inc.; a Montreal-based international firm specializing in software engineering and and software quality assurance training, management consulting and professional development. AmiBug helps companies set up software engineering and quality assurance teams and process through a combination of training and management consulting. Robert was the Director of Research and Development at Purkinje Inc where he was charged with developing world class critical medical software used by clinicians at the point of care. Previously, Robert managed Software Development at Alis Technologies for over ten years. He has built several successful software development teams and champions the implementation of "light effective process" to achieve excellence in delivering on-time, on-quality, on-budget commercial software solutions.

Robert has championed many complex international multilingual software

development and globalization efforts involving several intricate business partnerships and relationships including international government (Czech, Egypt, France, Morocco, Algeria...) and commercial entities (Microsoft, IBM, AT&T, HP, Thompson CSF, Olivetti...). Systems included concurrent coordinated multilingual multiplatform product releases.

Robert's pioneering work with Infolytica Corporation led to the development of the first commercially available platform independent graphics standard GKS and several toolkits which allowed for cross platform development and porting of complex CAD, Graphics, Analysis and Non-Destructive Simulation systems.

Robert is a frequent guest lecturer at McGill University where he relates theoretical aspects of Software Engineering to real world examples with practical hands-on demonstrations.

In 1999, Robert completed a short book illustrated by his daughter Catherine entitled "I Am a Bug" (ISBN 0-9685774-0-7). Written in the style of a children's book, "I am a Bug" explains elements of the software development process using a fun metaphor. Throughout his career, he has also been the author of several articles and papers, and has given presentations relating to software development at a number of international conferences. Robert presented an interactive half-day tutorial on Bug Priority and Severity at Software Quality Week in Belgium - November 1999, a half-day tutorial on becoming and effective SQA manager at Quality Week 2000 in San Fransisco, and frequently gives courses on Software Testing, Development related subjects.

Robert has received professional recognition for many accomplishments over the years. At TEPR 2000 - award for best electronic patient record product to EHS using the Purkinje CNC component. Byte Middle-East's 1992 Product of the Year for the AVT-710 product family achieving a ZERO FIELD REPORTED software defect rate with over 15,000 units installed. (Project involved over 27-man month's effort!); Quebec Order of Engineers' recognition for creating and managing the Alis R&D Policy Guide - Development Framework and process.

QWE2000 Tutorial G1

Mr. Adrian Cowderoy (ProfessionalSpirit)

"Cool Q- Quality Improvement For Multi-Disciplinary Tasks In Website Development"

Key Points

- Player identification. Identification of the business/marketing players. Use of profiling and stereotyping to model the great diversity that are common at most websites. Identification of the key content and domain experts who understand how to make the website effective for its primary business purpose. Identification of the tool features and legacy systems/content that constraint that website.
- Feature identification. Use of checklists, questionnaires, models and behaviour monitoring to identify the features required by the different players.
- Quality profiling to support constructive negotiation between the different players, and to assure a complete view of all important quality features.
- Commercial design. Achieving the balance between system constraints (technical quality) and system benefits. Design websites that are engaging, inspiring and (if you have the talent) cool.
- Killer-feature monitoring, to identify and track commercially sensitive system benefits.
- Use of size and complexity measures (and quality features) to identify components and design decisions that create high maintenance costs and risks for website content.
- Use of risk and cost assessment to optimize the quality profile and design complexity.
- Use of risk management planning to indicate how to improve quality in the project by appropriate staffing, testing, monitoring, and other quality-improvement actions.
- Creating a learning organization. Combining commercial training materials and in-house know-how to create a simple knowledge-base of good practice.

Presentation Abstract

The tutorial is targeted at website producer/directors and managers. It is also strongly recommended for software quality people who are moving into the Internet business. The tutorial addresses website content and structure, and the functionality resulting from using Internet development tools.

The tutorial begins by explaining the background of how the web embraces multi-disciplines and multi-practices, and how new web-savvy organizations go beyond the boundaries of traditional industries.

The tutorial continues by introducing a set of quality improvement methods that have been developed over the last three years, for web-site development. An explanation is provided of how these methods support each of the main archetypes that are used in the workplace. That is, the engineer, the hunter/trickster, the gatherer, and the warrior.

About the Speaker

Adrian Cowderoy is Managing Director of the Multimedia House of Quality Limited, a company which he established to promote quality-improvement methods for the production of websites and multimedia.

Mr Cowderoy was the General chair of ESCOM-SCOPE-99 and ESCOM-ENCRESS-98 conferences, and was Program chair for ESCOM 96 and 97 (The European Software Control and Metrics conference promotes leading-edge developments in industry and research, worldwide û see www.escom.co.uk). He is the METRICS-ESCOM Coordinator for IEEE METRICS 2001 and was on the Program committee of Metrics 98 and 99, European Quality Week 99 and COCOMO/SCM 96-99. In 1998 he was acting Conference Chair of the Electronics and Visual Arts conference in Gifu, Japan. He is a registered expert to the European Commission DGXIII.

He has provided consultancy and industrial training courses on quality management, risk management, and cost estimation to the aerospace and medical industries in the UK, Germany and Italy since 1995. He also lectures at Middlesex University (www.mdx.ac.uk) on e-commerce project management and managing Internet start-up's, and at City University, London (www.city.ac.uk), on project management for systems development.

Mr Cowderoy was project manager and technical director of MultiSpace, a 14-month million-dollar initiative sponsored by the European Commission in which 12 European organizations explored the potential to apply quality-improvement methods to multimedia and website development projects. (See www.mmhq.co.uk/multispace and www.cordis.lu/esprit.)

He was a Research fellow at City University from 1990-1998, and a Research Associate at Imperial College from 1986-1989. He was also a quality consultant and software developer at International Computers Limited, UK, from 1980-1985, where he worked on operating and networking systems for mainframes and distributed systems.

His academic qualifications include an MSc in Management Science from Imperial College, University of London in 1986, and is a member of the Association of MBA's. He received a BSc in Physics with Engineering from Queen Mary College, University of London, in 1979.

Mr. Cowderoy has published and presented extensively on multimedia quality and software cost estimation. He was joint editor of Project Control for 2000 and Beyond (Elsevier, 1998), Project Control for Software Quality (Elsevier, 1999), and Project

Control: The Human Factor (Elsevier, 2000).

QWE2000 Tutorial E2

Tom Gilb
(Result Planning Limited)

"Specification Quality Control (SQC): A Practical Inspection Workshop on Requirements Specification"

Key Points

- Point 1...
- Point 2...
- Point 3...

Presentation Abstract

A. Narrative Description 40-60% of all software bugs which escape test to the field user have been traced to requirements and design specifications before coding. It has then been proved that Inspecting the specifications sharply reduces this problem. This workshop will explore all aspects of Specification Quality Control in a hands on practical workshop. Participants will actively experience the technology necessary to attack this quality challenge.

B. Learning Objectives. 1. To learn the problem and the solutions mainly by means of personal experience. You will get a series of individual tasks which will teach you basic principles of specification and quality control of specification.

C. Detailed Outline 1. The Ambiguity Test: proving that specifications are unintelligible.

2. Rules: Selecting strong standards for specification which enable quality control.

3. Process Control: Deciding on the economically allowable Major Defect density allowed for a specification to be released to your colleagues

4. Planning: Selecting a suitable sample to check. Selecting checklists. Allocating specialist checking roles on the team. Deciding on checking rates using optimum rate data.

5. Checking: Individual effort to find defects, and especially Major defects.

6. Data Collection: Gathering data on the checking phase: defects, Majors, Rate, Sample size.

7. Extrapolation: Calculating probable team result of unique Major defects/ Logical Page. Calculating total defect density. Calculating defects remaining after corrections {per page, total}. Calculating total future rework costs based on remaining Major defects.

8. Drawing Conclusions: Can the document exit according to our Exit Conditions? If not, what should we do?

9. Observations. What did you learn? What surprised you the most? What do you think you should do back at your own work about these things? What barriers do you see to doing them? What can you do to remove the barriers?

About the Speaker

Tom Gilb is an independent consultant, teacher and author. He works mainly in UK, Europe and North America. He is resident in Norway.

Tom coined the term 'Software Metrics' with the publication of his book of the same name in 1976 (European edition) and 1977 (USA edition). This work is the acknowledged (by R. Radice and W. Humphrey) as inspiration for much of the Software Engineering Institute's Capability Maturity Model Level 4 (SEI CMM Level 4). His other books include Principles of Software Engineering Management (1988, now in 13th printing) and Software Inspection (1993 with Dorothy Graham). His main professional interest is the development of powerful Systems Engineering methods (covering Requirements, Design, Quality Control and Project Management)

QWE2000 Tutorial F2

Tom Drake
(Integrated Computer Concepts, Inc)

"The Quality Challenge for Network Based Software Systems"

Key Points

- Internet quality for network centric systems
- Enterprise testing
- Quality network systems theory

Presentation Abstract

This presentation would introduce a biologically inspired model-based conceptual framework for network-centric testing and quality assurance. It involves an architecture that can deal with computers and software viewed as a system of interactive and dynamic behavioral objects rather than strictly for data processing and number crunching that are themselves part of a larger system.

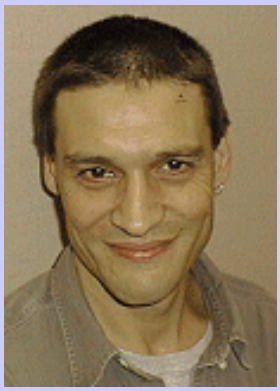
This conceptual framework for testing and quality assurance would allow for testing a range of behaviors and outcomes and the possible interactions for these application objects without the necessity for fully understanding them in advance! This could permit testing the fundamental structure of the program and the application environment and the executable functional mechanisms underneath as a testing and quality assurance framework that is anchored in living systems theory. It permits an inside out approach such that testing and quality engineering activities are based on the genetic makeup of the expected and anticipated dynamic state attributes and characteristics of the system using its own behavioral specifications as the test instruments for locating and stimulating the weak links.

This quickstart/tutorial presentation will also examine the significant challenges posed by network-based software systems and provide the context and background for understanding the daunting task faced by quality specialists and information technology management in dealing with this future frontier, today!

About the Speaker

Mr. Drake is a software systems quality specialist and management and information technology consultant for Integrated Computer Concepts, Inc. (ICCI) in the United States. He currently leads and manages a U.S. government agency-level Software

Engineering Center's quality engineering initiative. In addition, he consults to the information technology industry on technical management and software engineering and code development issues. As part of an industry and government outreach/partnership program, he holds frequent seminars and tutorials covering code analysis, software metrics, OO analysis for C++ and Java, coding practice, testing, best current practices in software development, the business case for software engineering, software quality engineering practices and principles, quality and test architecture development and deployment, project management, organizational dynamics and change management, and the people side of information technology. He is the principal author of a chapter on Metrics Used for Object-Oriented Software Quality for a CRC Press Object Technology Handbook published in December of 1998. In addition, Mr. Drake is the author of a theme article entitled: Measuring Software Quality: A Case Study published in the November 1996 issue of IEEE Computer. He also had the lead, front page article published in late 1999 for Software Tech News by the US Department of Defense Data & Analysis Center for Software (DACS) entitled: Testing Software Based Systems: The Final Frontier. Mr. Drake is listed with the International Who's Who for Information Technology for 1999, is a member of IEEE and an affiliate member of the IEEE Computer Society. He is also a Certified Software Test Engineer (CSTE) from the Quality Assurance Institute (QAI).



QWE2000 Tutorial G2

Mr. Tobias Mayer and Edward Miller (eValid, Inc.)

Website Testing

BACK TO QWE2000 PROGRAM

Key Points

- In the past few years web sites have grown from simple, static collections of HTML pages to complex pieces of software using advanced technologies including ASP, XML, script languages, e-commerce and more.
- Testing of such complexity is a forever-growing challenge.
- Testing of passive vs. interactive pages & static vs. dynamic pages will be explored.

Presentation Abstract

In the past few years web sites have grown from simple, static collections of HTML pages to complex pieces of software using advanced technologies including ASP, XML, script languages, e-commerce and more. Testing of such complexity is a forever-growing challenge. Testing of passive vs. interactive pages & static vs. dynamic pages will be explored. The use of a 'Test Enabled Web Browser' will be emphasised as the most effective way of realistically testing most web sites.

About the Speakers

Tobias Mayer is a senior software engineer at Software Research, Inc. He is reponsible for the main design and implementation of the "eValid" Web Test engine. Tobias has a (UK) BSc from South Bank University, London. He is a member of, and OO Metrics consultant to, the Center for Systems & Software Engineering (CSSE) at South Bank University. Tobias has presented and published a number of papers on OO metrics, including papers at IEEE 'TOOLS' 1999 and British Computer Society 'SQM' 1999. During this year, Tobias has presented a number of seminars on Website Testing strategies in the UK. He also presented the "Quickstart - Website Testing" seminar at the 'Quality Week 2000' conference in San Francisco, June 2000.

Dr. Edward Miller is President of Software Research, Inc., San Francisco, California, where he has been involved with software test tools development and software engineering quality questions. Dr. Miller has worked in the software quality management field for 25 years in a variety of capacities, and has been involved in

the development of families of automated software and analysis support tools.

He was chairman of the 1985 1st International Conference on Computer Workstations, and has participated in IEEE conference organizing activities for many years. He is the author of Software Testing and Validation Techniques, an IEEE Computer Society Press tutorial text. Dr. Miller received his Ph.D. (Electrical Engineering) degree from the University of Maryland, an M.S. (Applied Mathematics) degree from the University of Colorado, and a BSEE from Iowa State University.

BACK TO QWE2000 PROGRAM

QWE2000 Keynote Session K1-1

Tom Gilb
(Results Planning Limited)

The Ten Most Powerful Principles for Quality in Software Organizations

Presentation Abstract

Software knows it has a problem. Solutions abound. But which solutions work? What are the most fundamental underlying principles we can observe behind those successful solutions? Can these principles guide us to select successful solutions and avoid time wasters? One hint: in observing successful software organizations in the US, the dominant principle seems to be feedback and control.

About the Speaker

Tom Gilb is an independent consultant, teacher and author. He works mainly in UK, Europe and North America. He is resident in Norway.

Tom coined the term 'Software Metrics' with the publication of his book of the same name in 1976 (European edition) and 1977 (USA edition). This work is the acknowledged (by R. Radice and W. Humphrey) as inspiration for much of the Software Engineering Institute's Capability Maturity Model Level 4 (SEI CMM Level 4). His other books include Principles of Software Engineering Management (1988, now in 13th printing) and Software Inspection (1993 with Dorothy Graham). His main professional interest is the development of powerful Systems Engineering methods (covering Requirements, Design, Quality Control and Project Management)

"The Ten Most Powerful Principles for Quality in [Software and] Software Organizations"

Quality Week Europe, Brussels

Wednesday 22 November 2000 0830-1000

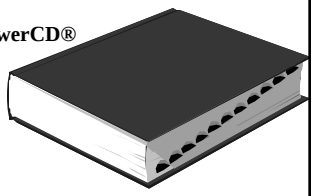
ABSTRACT:
Software knows it has a problem. Solutions abound. But which solutions work? What are the most fundamental underlying principles we can observe behind those successful solutions? Can these principles guide us to select successful solutions and avoid time wasters? One hint: in Observing successful software organizations in the US, the dominant principle seems to be feedback and control.

By Tom Gilb, Result Planning Limited

Gilb@acm.org, www.result-planning.org

"Principle" is ...

Webster's New World™ College Dictionary (Third Edition) on PowerCD®



prin•ci•ple (prinse pel)

n.

1 the ultimate source, origin, or cause of something

2 a natural or original tendency, faculty, or endowment

3 a fundamental truth, law, doctrine, or motivating force, upon which others are based [moral principles]

4 a) a rule of conduct, esp. of right conduct b) such rules collectively c) adherence to them; integrity; uprightness [a man of principle]

5 an essential element, constituent, or quality, esp. one that produces a specific effect [the active principle of a medicine]

6 a) the scientific law that explains a natural action [the principle of cell division] b) the method of a thing's operation [the principle of a gasoline engine is internal combustion]

in principle

theoretically or in essence

on principle

because of or according to a principle

Slide **3**

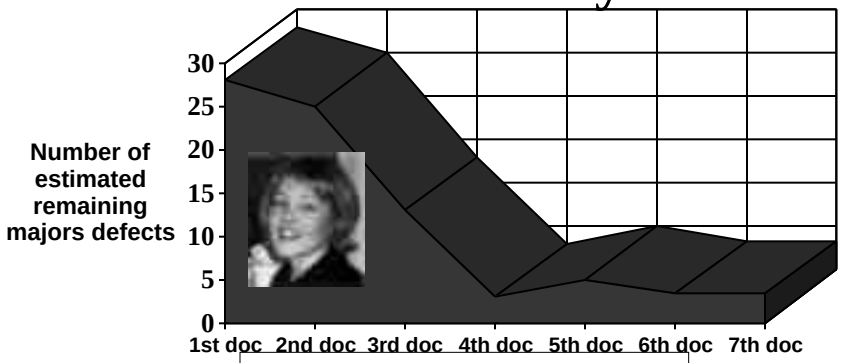
1. Feedback

- Rapid feedback allows rapid correction.
  - Methods using rapid feedback succeed, those without seem to fail.
    - Methods:
      - Defect Prevention Process (CMM 5, Mays, IBM 1985)
      - Inspection (Fagan, IBM 1975) *
      - Evolutionary Project Management (Mills, IBM, Cleanroom, 1970) *
      - Statistical Process Control (SPC): Shewhart, Deming, Juran (1920's)

* reprints are in IBM Systems Journal, 2&3 1999

Slide **4**

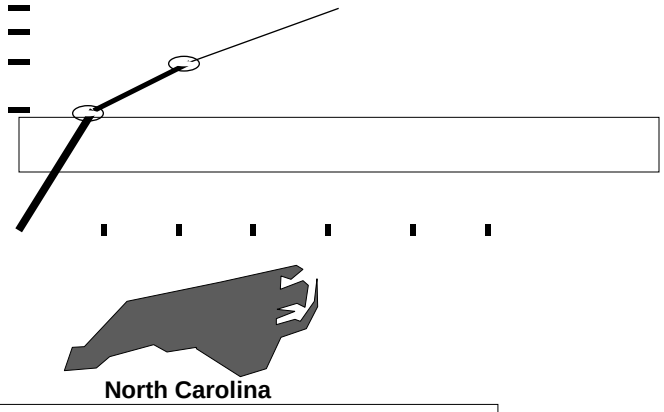
Marie's Learnability Curve



Number of Document Inspections where she got useful feedback about quality and rules.

Slide **5**

Defect Prevention Experiences: Most defects can be prevented from getting in there *at all*



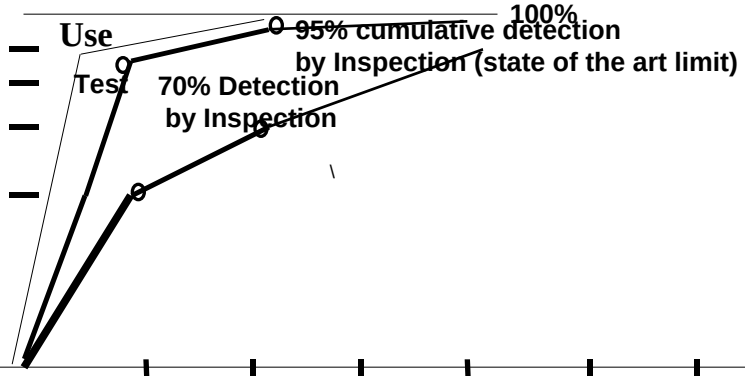
IBM Research Triangle Park Networking Laboratory

5

Gilb@acm.org, 2000

Slide **6**

Prevention + Pre-test Detection is the most effective and efficient



- Prevention data based on state of the art prevention experiences (IBM RTP), Others (Space Shuttle IBM SJ 1-95) 95%+ (99.99% in Fixes)
- Cumulative Inspection detection data based on state of the art Inspection (in an environment where prevention is also being used, IBM MN, Sema UK, IBM UK)

Slide **7**

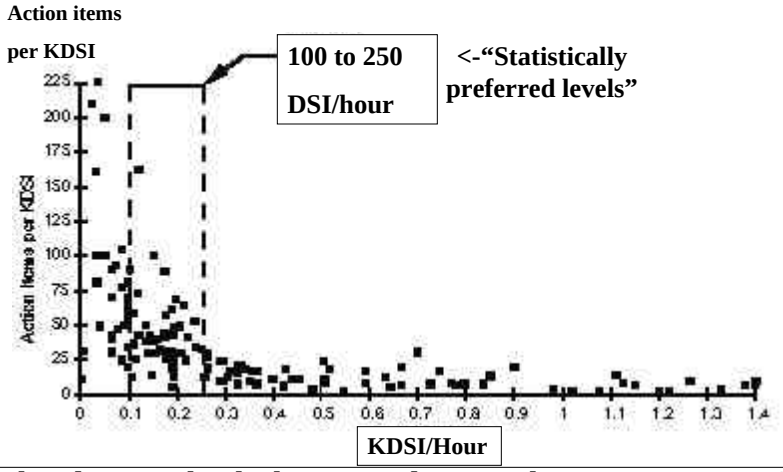
IBM MN & NC DPP Experience.
High quantity feedback leads to real change.

- **2162 DPP Actions implemented**
  - **between Dec. 91 and May 1993 (30 months) <-Steve Kan**
- **RTP about 182 per year for 200 people. <-Robert Mays 1995**
  - **1822 suggested ten years (85-94)**
  - **175 test related**
- **RTP 227 person org <- Mays slides**
  - **130 actions (@ 0.5 work years**
  - **34 causal analysis meetings @ 0.2 work years**
  - **19 action team meetings @ 0.1work years**
  - **Kickoff meeting @ 0.1 work years**
  - **TOTAL costs 1% of org. resources**
- **total ROI (Return On Investment) DPP 10:1 to 13:1,**
- **internal ROI 2:1 to 3:1**
- **Defect Rates at all stages 50% lower with DPP**

Fault Density versus Checking Rate: Raytheon 95

Slide **8**

Feedback on optimum rates leads to orders of magnitude better performance

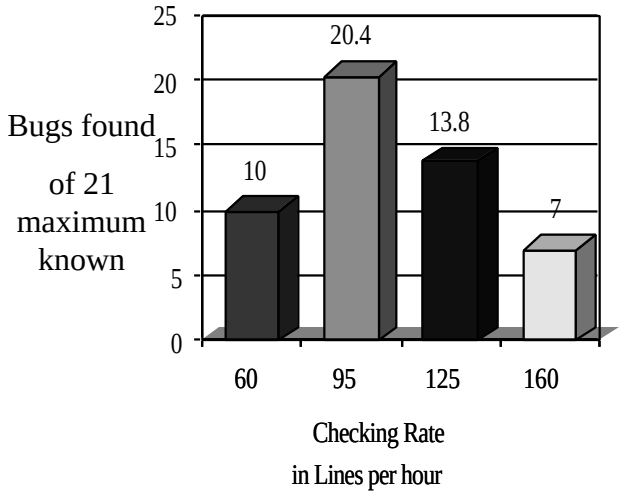


Why do you think they avoid using the optimum rate?

Hint: "Our process mandates 100% inspection coverage"

Effectiveness a function of checking rate (Buck)
Feedback on optimum rates permits more effective performance

Slide **9**

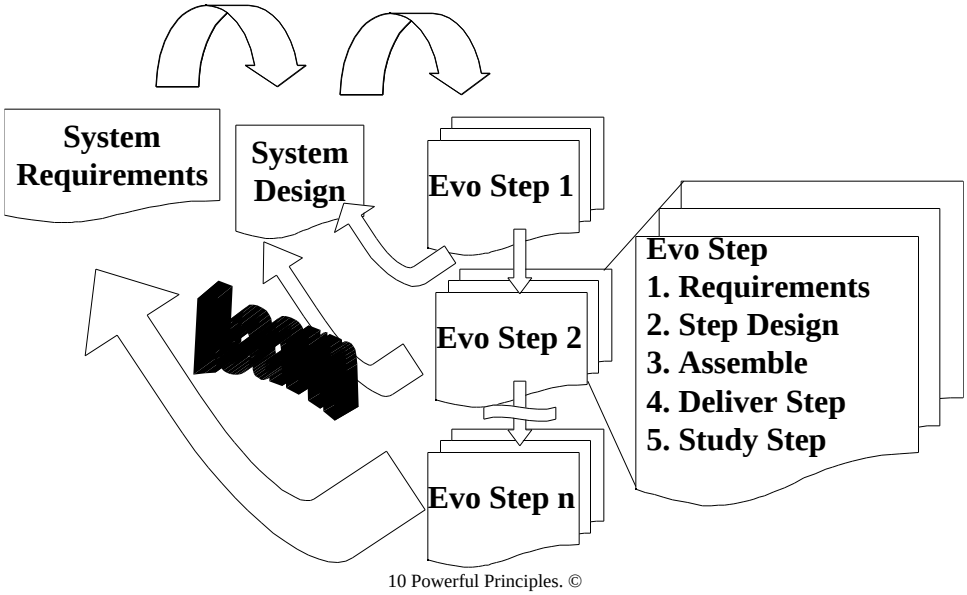


From Frank Buck IBM 1980

Evo 'Learning' model

Slide **10**

Project feedback improves requirements and design and process!



Evo shortens project by feedback (MS)

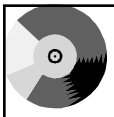
- "It appears that this incremental approach takes longer, but it almost never does, because it keeps you in close touch with where things really are"
- Brad Silverberg, Sr. VP for Personal Systems Microsoft in CUSUMANO95 , page 202

Customer feedback weekly!

An example of a typical one-week Evo cycle at the HP Manufacturing Test Division during a project. [MAY96, HP* Journal Aug 96]

	Development Team	Users
Monday	✓ System Test and Release Version N ✓ Decide What to Do for Version N+1 ✓ Design Version N+1	
Tuesday	✓ Develop Code	✓ Use Version N and Give Feedback
Wednesday	✓ Develop Code ✓ Meet with users to Discuss Action Taken Regarding Feedback From Version N-1	✓ Meet with developers to Discuss Action Taken Regarding Feedback From Version N–1
Thursday	✓ Complete Code	
Friday	✓ Test and Build Version N+1 ✓ Analyze Feedback From Version N and Decide What to Do Next	

* one of my direct customers, TG

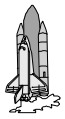


Direct Customer Input (MS)

- "Microsoft's general philosophy has been to ….. focus on evolving features and whole products incrementally, with *direct input from customers during* the development process."
  CUSUMANO95 , 13, Microsoft Secrets

Harlan Mills on Project Control:
2% deliveries feedback gives full project control!

- **"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, from 1996 a part of Lockheed Martin Marietta) "some ten years ago [about 1970] in a continuing evolution that is still underway.**
  - **Ten years ago general management expected the worst from software projects – cost overruns, late deliveries, unreliable and incomplete software.**
  - **Today [1980] , management has learned to expect on-time, within budget, deliveries of high-quality software.**
- **A Navy helicopter ship system, called LAMPS, provides a recent example.**
  - **LAMPS software was a four-year project of over 200 person-years of effort,**
  - **developing over three million, and integrating over seven million words of program and data for eight different processors distributed between a helicopter and a ship,**
  - **in 45 incremental deliveries.**
  - **Every one of those deliveries was on time and under budget.**
- **A more extended example can be found in the NASA space program,**
  - **where in the past ten years, FSD has managed some 7,000 person-years of software development, developing and integrating over a hundred million bytes of program and data for ground and space processors in over a dozen projects.**
  - **"There were few late or overrun deliveries in that decade, and none at all in the past four years." Harlan Mills [IBM Systems Journal No. 4, 1980, p. 415], Reprinted IBM SJ Vol. 38 1999, 289-295**



User Feedback (JPL)

- Evo “expects active feedback from the experience gained from one incremental delivery to the requirements from the next.
- As Evo periodically delivers to the users an increment of capability, the users are able to provide understanding of how effectively that delivery is meeting their needs.
- As the users assess the impact of a delivery on their operations, the system developer is able to work with them to adjust the system requirements to better satisfy their operational needs.
- Evo lets that adjusted set of requirements be the basis for all subsequent incremental deliveries.
- This feedback process is formal and proactive. It is a key element in making Evo effective from a user’s perspective.”
- [SPUCK93] Jet Propulsion Labs

2. Critical Measurement

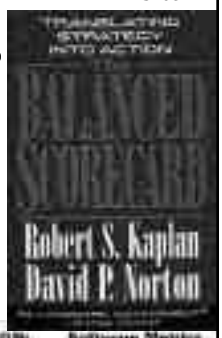
- If you do not focus on the few measures critical to your system, then it will fail.

- *This principle is supported by the slide detail for several other principles here, so I will not comment in more detail just here. TG*

Slide **17**

3. Multiple Objectives

- If you cannot control multiple measures of quality and cost simultaneously, then your system will fail due to the ones you did not control.



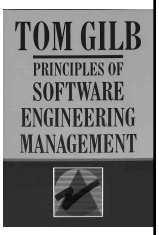
10 Powerful Principles. © Gilb@acm.org, 2000

Impact Table for Step Management: how to directly control many cost and quality objectives in small evolutionary project steps simultaneously **18**

	Step #1 Plan A: {Design-X, Function-Y}	**Step #1 Actual**	**Difference. - is bad + is good**	**Total Step 1**	**Step #2 Plan B: {Design Z, Design F}**	**Step #2 Actual**	**Step #2 Difference**	**Total Step 1+2**	**Step #3 Next step plan**
Reliability 99%-99.9%	**50% ±50%**	**40%**	**-10%**	**40%**	**30% ±20%**	**20%**	**-10%**	**60%**	**0%**
Performance 11sec.-1 sec.	**80% ±40%**	**40%**	**-40**	**40**	**30% ±50%**	**30%**	**0**	**70%**	**30%**
Usability 30 min. -30 sec.	**10% ±20%**	**12%**	**+2%**	**12%**	**20% ±15%**	**5%**	**-15%**	**17%**	**83%**
Capital Cost 1 mill.	**20% ±1%**	**10%**	**+10%**	**10%**	**5% ±2%**	**10%**	**-5%**	**20%**	**5%**
Engineering Hours 10,000	**2% ±1%**	**4%**	**-2%**	**4%**	**10% ±2.5%**	**3%**	**+7%**	**7%**	**5%**
Calendar Time	**1 week**	**2 weeks**	**-1week**	**2 weeks**	**1 week**	**0.5 weeks**	**+0.5 wk**	**2.5 weeks**	**1 week**

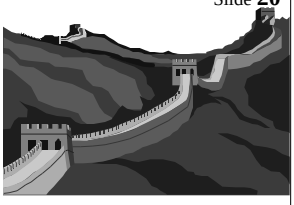
4. Evolution

- You must evolve in small steps towards your goals; large step failure kills the entire effort.
  - And early frequent result delivery is politically and economically wise.
  - 2% of total is a small step, you can afford to fail on



Ericsson case in Japan

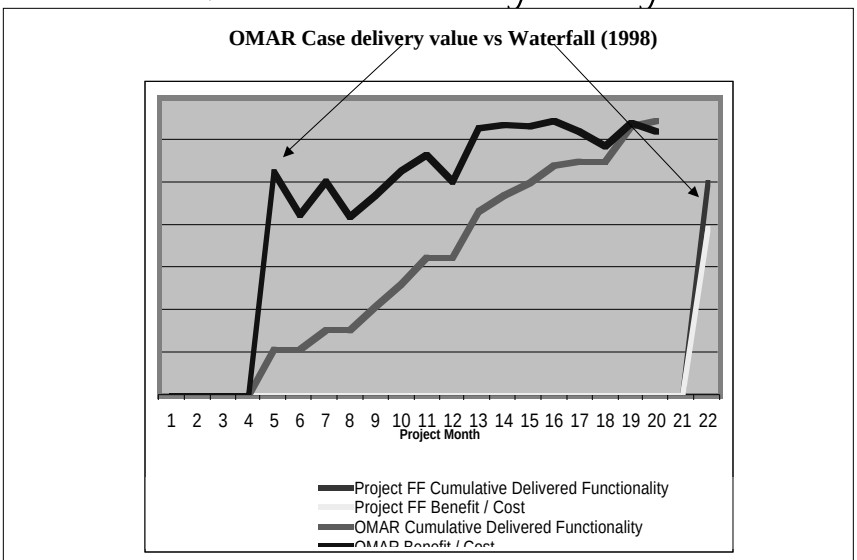
Tao Te Ching (500BC)



- That which remains quiet, is easy to handle.
- That which is not yet developed is easy to manage.
- That which is weak is easy to control.
- That which is still small is easy to direct.
- Deal with little troubles before they become big.
- Attend to little problems before they get out of hand.
- For the largest tree was once a sprout,
- the tallest tower started with the first brick,
- and the longest journey started with the first step.
  - From Lao Tzu in Bahn, 1980 Penguin book

Slide **21**

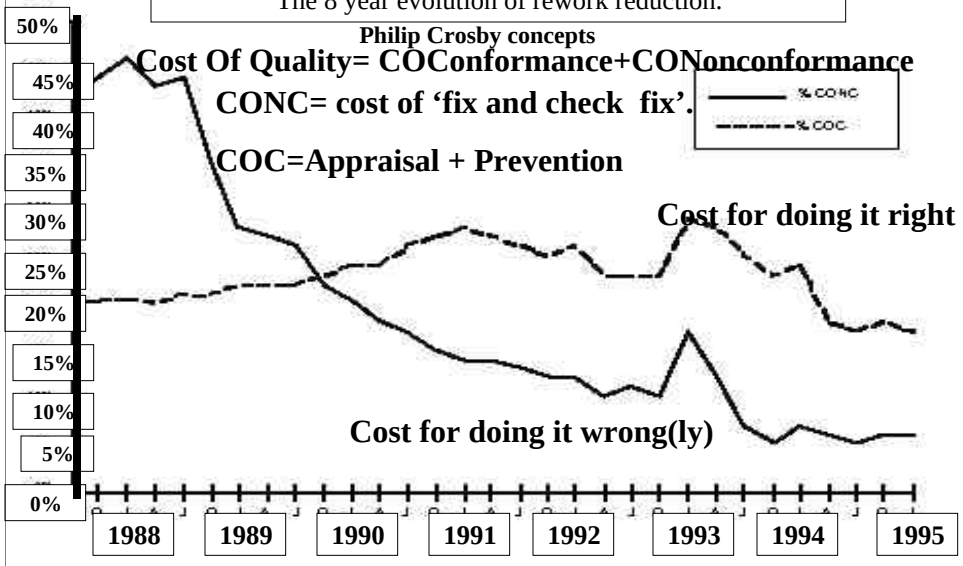
Value delivery early



Stuart Woodward: Evolutionary project Management
IEEE Computer Oct 1999, page 49-57
s.woodward@computer.org

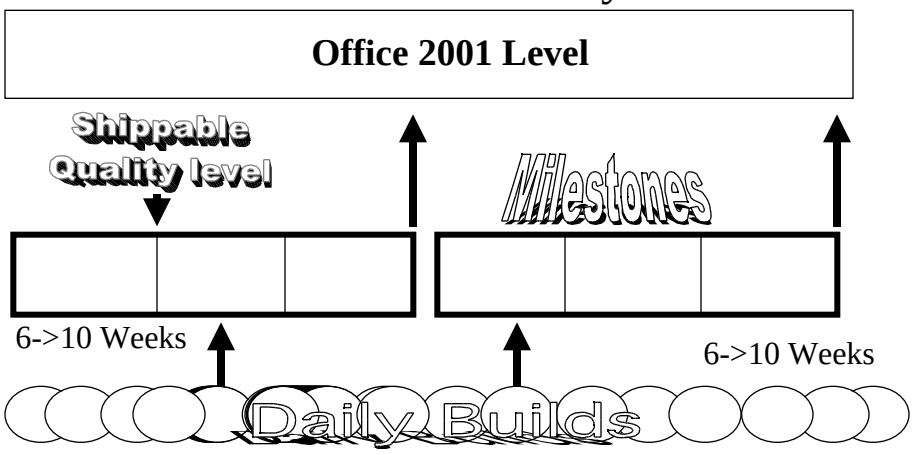
Slide **22**

Cost of Quality versus Time: Raytheon 95
The 8 year evolution of rework reduction.



Project Cost = {Cost of Quality + Cost of Performance}.
Cost of Performance={Planning, Documentation, Specification}.

Multiple Levels of Microsoft Evo: Ms does Evo daily!



Go to next Graphic

Early simple proof of concept (Ericsson):

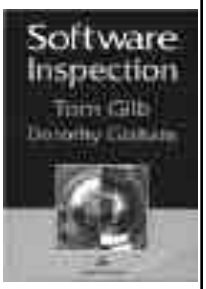
Ericsson used Evo to deliver a 15 month project in 9 months to Japan

ERICSSON

- **"Organic integration [Evo] is a way of getting rid of the myth [that problems don't exist] very early on.**
- **You could say that organic integration demands of an organization that it do the specifications, the system, the design and the verification for *one first very small task* very quickly.**
- **It also demands of the organization that it *do this right* in terms of delivering products correctly.**
- **If the organization cannot even manage its first simple task in the time agreed, it certainly should *question the ability* to manage more difficult tasks.**
- **This *process of questioning* is very healthy. It may for example prevent the *delusions of grandeur* so common in nearly all organizations".**
- **[Ericsson94], page 26, Jack Järkvik, in the context of building mobile telephone base stations**

Ericsson case in Japan

5. Quality Control



- Quality Control must be done as early as possible, in planning, to reduce the delays from late defect finding.
  - Use numeric Exit from development process
    - Like "Maximum 0.2 Majors/Page"
  - Use Inspection sampling to keep costs down, and to permit early, before completion, action and learning.

August 1999

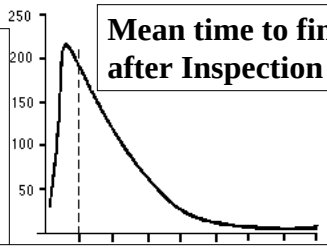
10 Top Advanced Inspection Principles

- Pr1. *Prevention* is more effective than Cure
- Pr2. *Avoidance* is more efficient than removal
- Pr3. *Feedback* teaches effectively
- Pr4. *Measurement* gives facts to control the process
- Pr5. Priority to the *Profitable*
- Pr6. *Forget perfection*, you can't afford it!
- Pr7. Teach *fishing*, rather than 'give fish'
- Pr8. *Framework for Freedom* beats bureaucracy
- Pr9. *Reality* rules
- Pr10. *Facts* beat intuition

Slide **27**

The downstream alternative cost of quality at a Defence Electronics Factory (all types of documents).

Number of defects of the 1,000 sampled Majors



Estimated hours to find and correct in test or in field. Range 1 to 80 hours

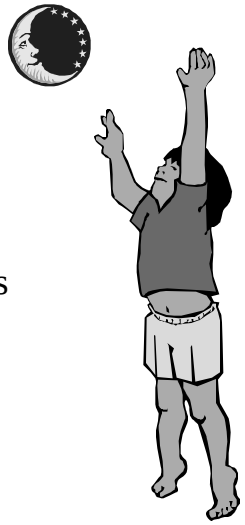
Mean time to find and correct a Major after Inspection was 9.3 Hours.

It cost about 1 hour to find and fix a Major using Inspection

Slide **28**

Advanced Inspection Objectives

- Central Objectives
  - 1. Engineering Process Control
  - 2. *Measuring* Document Quality
  - 3. Reduce Project Time & Cost
- Secondary Objectives
  - 4. Identify and Remove Major Defects
  - 5. Reduce Service/Maintenance Costs
- *NOT* Objectives
  - Approve document '*content*'
  - Remove *minor* defects
  - '*Improve*' Quality



Larger set of Inspection Objectives

1. Time-to-Delivery
2. Measurement
   - •document quality
   - •doc. process quality
   - •inspection value/cost
3. Release "downstream"
4. Identify defects
5. Fix defects
   avoid new defect injection
6. Improve process
   product producers
   inspection itself
7. On-the-job training

8. Motivation
9. Help Author
10. Effectiveness (Quality)
11. Efficiency (Productivity)
12. Train Inspection team leader
13. Certify the leader
14. Motivate Managers
15. Reduce Maintenance Costs
16. Relieve Project Leader.
17.many others

Slide **30**

6. Motivation

- The 'best methods' work only when people are motivated
  - 'Drive out fear' (Deming)

Slide **31**

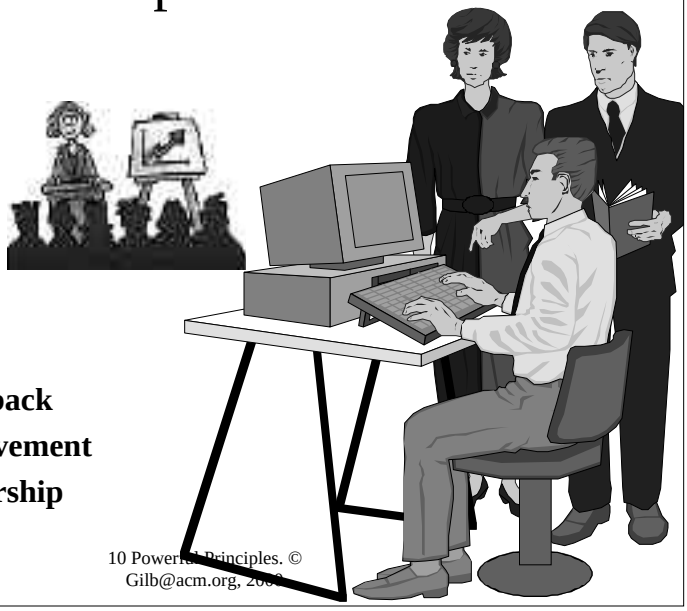
Motivation ‘is Everything!’

- **People are ‘sensitive’**
- **Avoid all ‘threats’**
- **Give ‘positive’ motivators**
- **Very many ‘details’ support this attitude**
- **You will respect this, or fail!**
- **Do unto others, as you would have them do with *your* work!**

Slide **32**

Positive Motivators in our Inspection version

- **Group-work**
- **Team**
- **Freedom**
- **Learning**
- **Game**
- **Experiments**
- **Challenge**
- **Numeric Feedback**
- **Process Improvement**
- **Positive Leadership**
- **Sampling**



Slide **33**

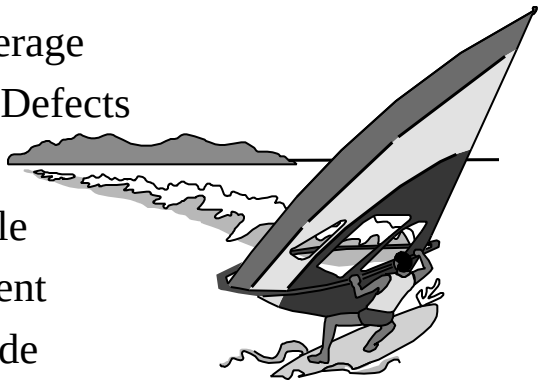
Potentially Negative Motivators in bad Inspection practice

- Time Pressure
- Result Pressure
- Personal Attacks
- Bureaucracy
- Small-minded Leader
- Personal-fault blaming
- Process corruption
- High volume/cleanup

Slide **34**

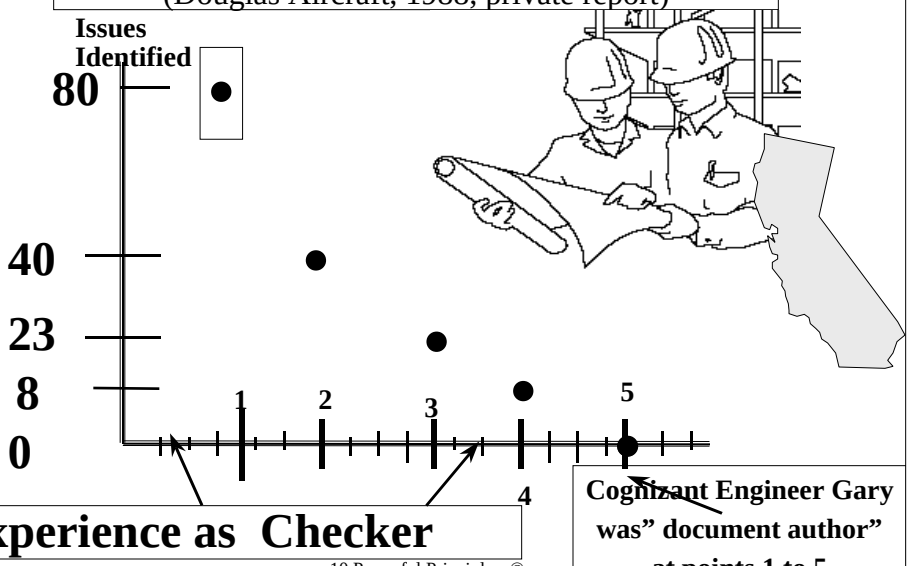
Motivational Philosophy

- Intelligent Inspection
- Maximum Leverage
- Process causes Defects
- Trust people
- Empower people
- Allow experiment
- Let results decide
- Continuously improve



Slide **35**

Gary's Personal Learning
(to follow process) Curve
(Douglas Aircraft, 1988, private report)



Slide **36**

MOTOROLA
Motorola India Electronics Ltd.

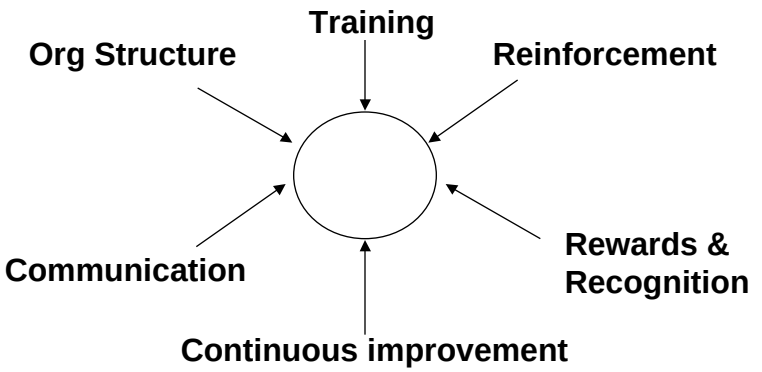
THE BUSINESS IMPACT OF REACHING CMU/SEI LEVEL 5

Roger G. Fordham
Ex Managing Director

"The Senate", No. 33A, Ulsoor Road
Bangalore-560 042, India
Ph: +91-80-559-8866; Fx: +91-80-559-8843
Visited by Gilb March 2000

Slides July 1996 Given Gilb by Fordham 1999

MIEL ~ THE EXPERIMENT



ORGANIZATIONAL VALUE SYSTEM & MANAGEMENT COMMITMENT

CONCLUSION

- ***Benefits of a well-controlled process in terms of quality, productivity and cycle time are very apparent.***
- ***Developing software across an ocean can be done in no other better way.***
- ***Process maturity provides a sense of self-esteem for individuals.***
- ***Process ownership has to lie with the decision makers.***
- ***Complete commitment, cooperation & participation from all levels of management required.***

CONCLUSION(2)

- ***Process maturity requires an open & mature culture.***
- ***Fear of making / admitting failures should not exist, however all failures should provide lessons learnt & same mistakes should not be repeated.***
- ***Involvement wears out resistance.***
- ***Empowerment is key to process maturity. It must be tempered with explicit bounds on what employees can & cannot address.***
- ***Long term cost benefit orientation will help in directing organizational change.***

RESULTS (1 of 2)

- LINES OF CODE RELEASED IN 1995
  - OVER 3 MILLION
- PRODUCTIVITY
  - 2 TIMES THE INDUSTRY AVERAGE
- POST-RELEASE QUALITY
  - **190 TIMES** INDUSTRY AVERAGE*
    - (they had 2 bugs in 800,000 LOC!, TG)
- 85% OF PROJECTS ARE DELIVERED ON SCHEDULE
- CUSTOMER SATISFACTION HAS BEEN CONSISTENTLY BETWEEN GOOD & EXCELLENT.

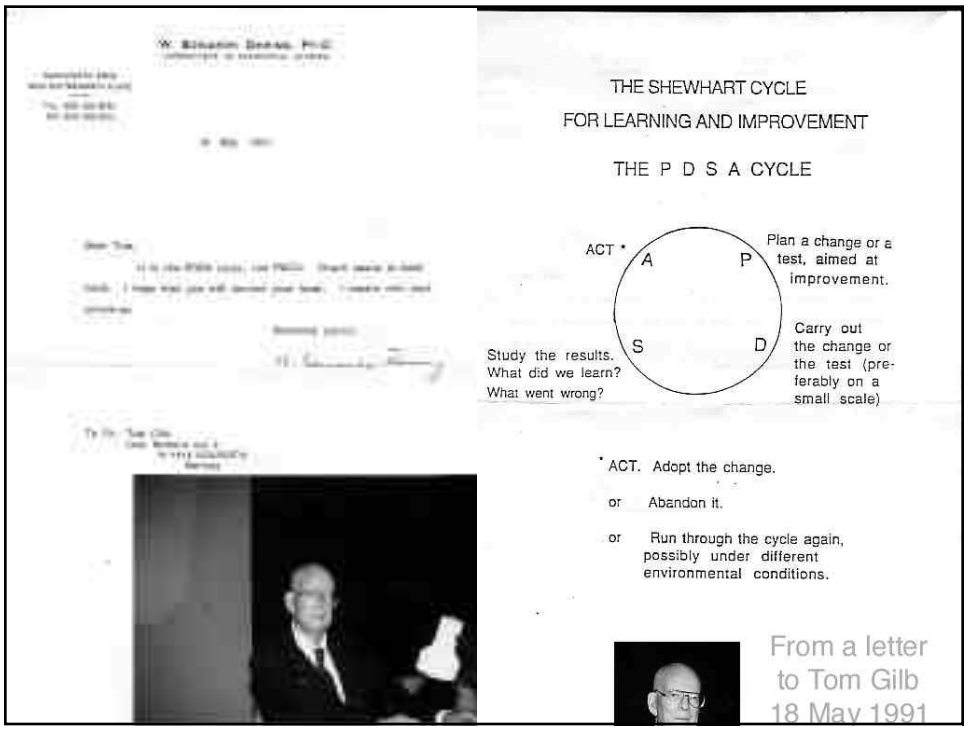
* US AVERAGE POST RELEASE DEFECTS OF 0.75 DEFECTS/FUNCTION POINT
(6 DEFECTS/1024LOC) *Industry average*
SOURCE: CAPERS JONES

RESULTS (2)

- BUILT BASELINES OF PRODUCTIVITY & DEFECT DENSITY FOR ELEVEN CATEGORIES OF PROJECTS.
- HAVE ACHIEVED BETTER THAN 20% ACCURACY FOR DEFECT PROJECTIONS 50% OF THE TIME
- BUILT SUFFICIENT HISTORICAL DATA FOR A BETTER REFINEMENT OF THE REGRESSION MODEL.
- BUSINESS HAS GROWN 300% IN THE LAST 5 YEARS.

7. Process Improvement

- Eternal Process improvement is necessary as long as you are in competition
  - Paraphrasing Deming about PDSA cycle end.



Process Brainstorming

The Road To Wisdom : Piet Hein

"The road to wisdom

is plain and simple to express,

to err, and err, and err again,

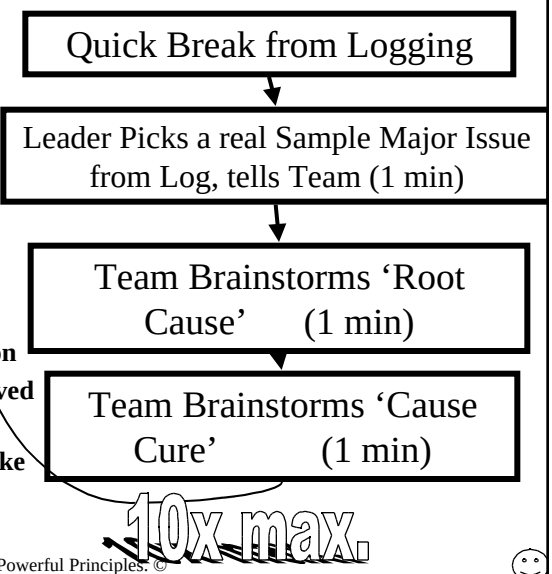
but,

less, and less, and less."

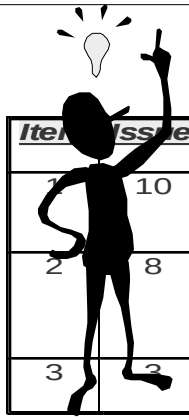
Piet Hein,
(Danish Philosopher

The PB Process

- **Team Stays together after 'Logging'**
  - **Same room**
  - **Same people**
  - **Maybe a break first**
  - **Same documents**
  - **Up to half an hour**
- **Shift mentality!**
  - **Not the project**
  - **The process, our organization**
  - **How we feel it can be improved for us**
  - **So we are not 'forced' to make mistakes**



The P.B. Log

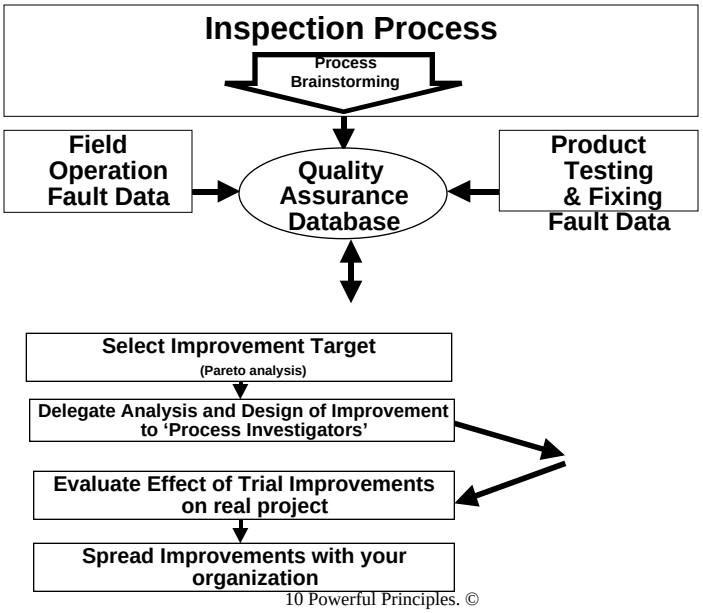


Item	Issue	Classify	Root Causes	Improvement Suggestions
1	10	Oversight	• Time pressure • no tools • no info	• optimum time • build tool • give info on PC
2	8	Education	• trainees don't know • manual not updated	• special meeting for trainees • manual on Web
3	3	Commun-ication	• authors are unknown	• publish their email on doc. head

- Brainstorming Rules: no criticism, flow ideas in
- Getting 'Grass Roots' opinions, investigation later

Defect Prevention Process within 'Inspection'

Slide **47**



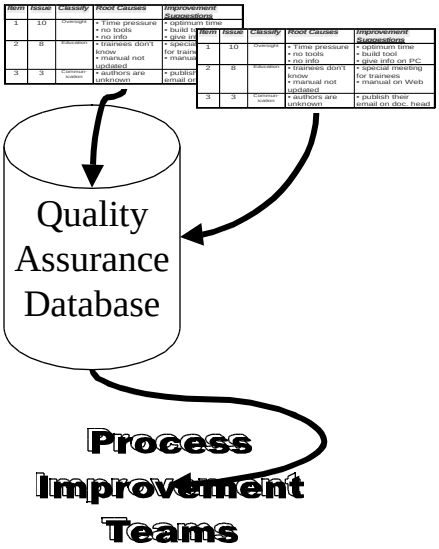
See 'Software Inspection' Chapters 7 and 17 for detail.

The Process Brainstorming Aftermath

Process Brainstorming

Slide **48**

- **Brainstormed suggestions**
  - **Are input to Process Improvement Teams.**
  - **Are part of the inputs**
    - **& cost of defect data**
    - **& frequency of defect.**
  - **PB Insights are**
    - **Accurate**
    - **Decentralized**
    - **Real time**
    - **Socially acceptable**
    - **Proven (Mays) to work better than centralized efforts (Fagan's Method 1973)**



Slide **49**

8. Persistence

- Years of persistence are necessary to change a culture.
- W. Edwards Deming
  - It takes 2-3 years to change a project, and a generation to change a culture
- Piet Hein (Denmark)
  - Things Take Time (TTT)

Slide **50**

1 $ $ $

IBM®

Secrets of Software Quality

Software Quality Week

Craig Kaplan, Ph.D.

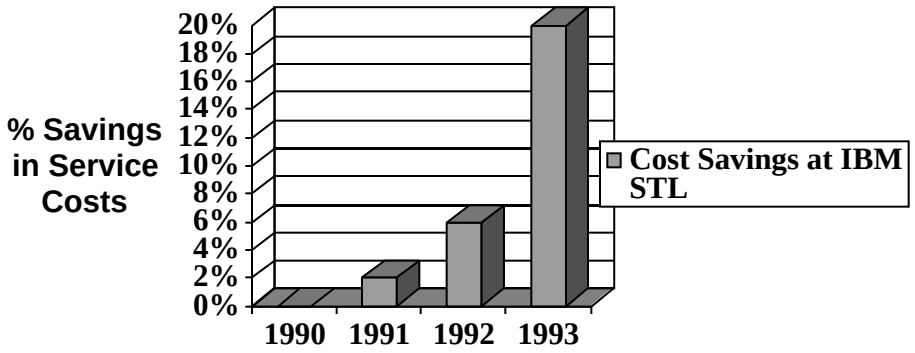
ckaplan@iqco.com

I.Q. Company

http:\\www.iqco.com

Slide **51**

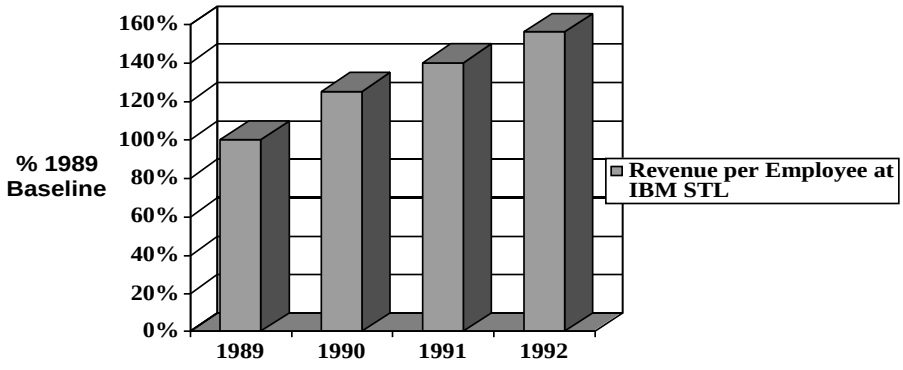
20% Savings on Service Costs
Note the 4 year time perspective.



Source: Secrets of Software Quality by Kaplan, Clark, & Tang (McGraw-Hill 1995)

Slide **52**

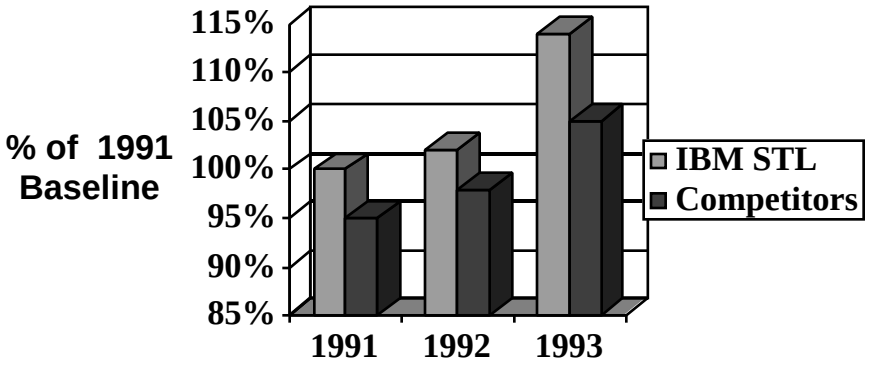
56% Increase in
Revenue per Employee



Source: Secrets of Software Quality by Kaplan, Clark, & Tang (McGraw-Hill 1995)

Slide **53**

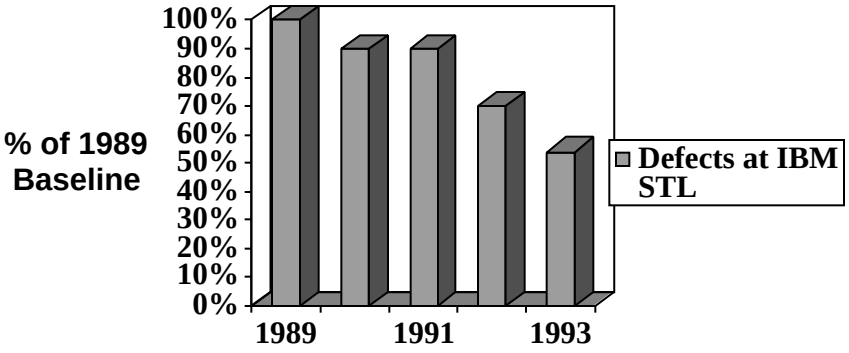
14% Improvement in Customer Satisfaction



Source: Secrets of Software Quality by Kaplan, Clark, & Tang (McGraw-Hill 1995)

Slide **54**

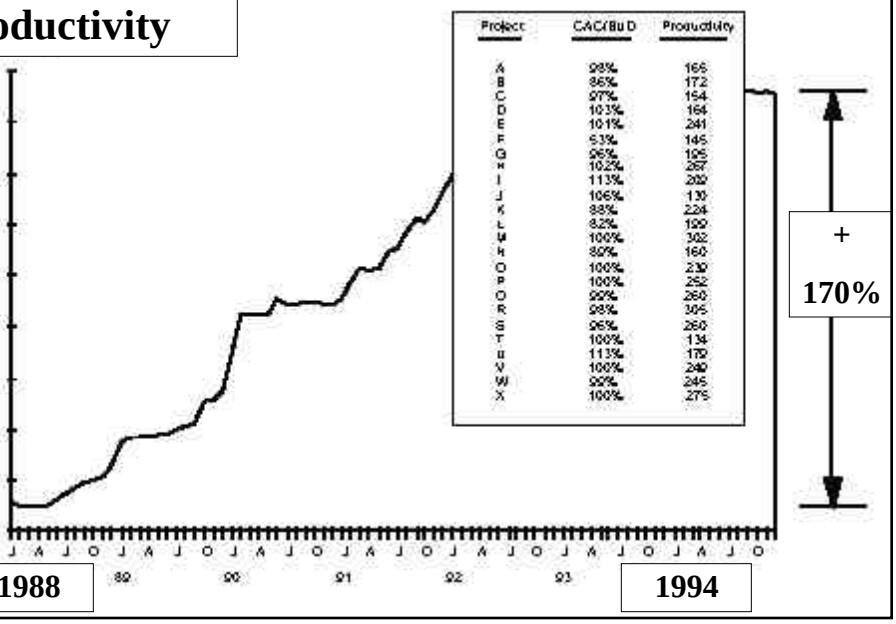
46% Reduction in Field Defects



Source: Secrets of Software Quality by Kaplan, Clark, & Tang (McGraw-Hill 1995)

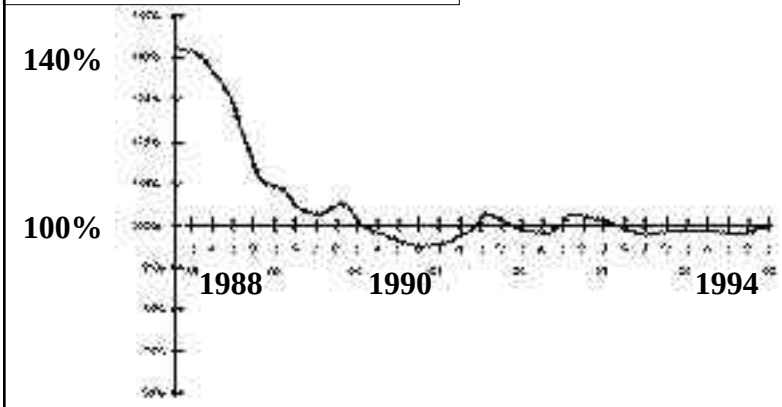
Raytheon 95 Software Productivity 2.7X better.
Note the 5 years to peak time perspective.

Productivity

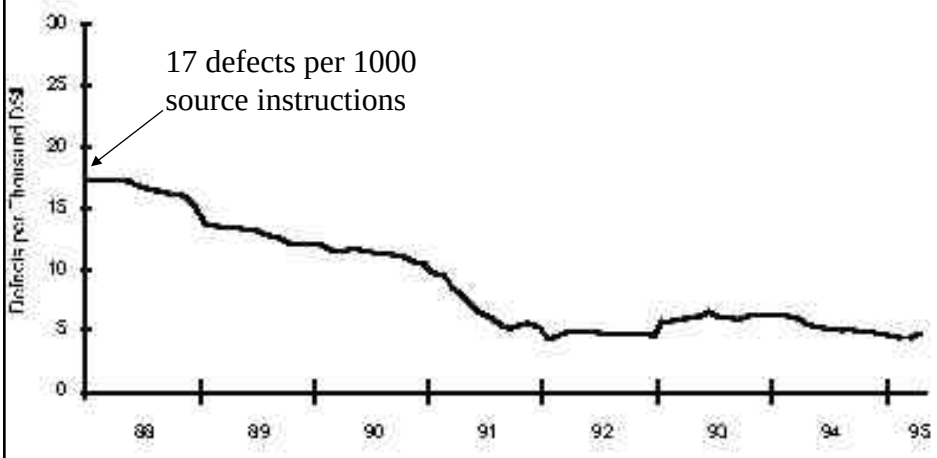


Achieving Project Predictability: Raytheon 95:
This miracle took almost 2 years

Cost At Completion / Budget %



Overall Product Quality: Raytheon 95
Defect Density Versus Time: 3 times better.
This took about 4 years.



Source:http://www.sei.cmu.edu/products/publications/95.reports/95.tr.017.html

9. Multiple Impacts

- Any method you choose will have multiple quality and cost impacts, whether you like them or not!
  - We need to estimate all impacts on our objectives
  - We need to reduce or accept negative impacts
  - We must avoid simplistic one-dimensional arguments

Slide **59**

Single next Step Comparison Table
Evaluating multiple impact to decide which step to deliver first.

	Next-Step Candidate **A**: {Design-X, Function-Y}	Next-Step Candidate **B:** {Design Z, Design F}
Reliability 99%-99.9%	**50%**	100%
Performance 11sec.-1 sec.	**80%**	30%
Usability 30 min.-30 sec.	**-10%**	20%
Capital Cost 1 mill.	**20%**	5%
Engineering Hours 10,000	**2%**	10%
Performance/Capital Cost Ratio	80/20= *4.0*	30/5= *6.0*
Quality/Cost Ratio	**120/22=5.46**	150/15=**10.00**

For written details of Impact Estimation method: see Competitive Engineering, free at www.result-planning.org and available from Addison Wesley

Slide **60**

Risk Analysis for each Step
Which is 'best' when risk is considered, on multiple qualities and costs?

	Step Candidate A: {Design-X, Function-Y}	**Step Candidate B: {Design Z, Design F}**
Reliability 99%-99.9%	**50% ±50%**	**100% ±20%**
Performance 11sec.-1 sec.	**80% ±40%**	**30% ±50%**
Usability 30 min.-30 sec.	**-10% ±20%**	**20% ±15%**
Capital Cost 1 mill.	**20% ±1%**	**5% ±2%**
Engineering Hours 10,000	**2% ±1%**	**10% ±2.5%**
Worst Case B/C ratio (1 to 3)	**(0+40-30)/(21+3) =*0.42***	**(80-20+5)/(7+12.5) =*3.33***
Best Case B/C ratio	**(100+120+10)/(19+1) = 11.5**	**(120+80+35)/(3+7.5)= 22.38**

Slide **61**

Step Choice with 'Credibility'

Evaluating multiple impacts with respect to experience spread and evidence credibility.

	Step Candidate **A**: {Design-X, Function-Y}	Step Candidate **B:** {Design Z, Design F}
Reliability 99%-99.9%	50% ±50%	100% ±20%
Performance 11sec.-1 sec.	80% ±40%	30% ±50%
Usability 30 min.-30 sec.	-10% ±20%	20% ±15%
Capital Cost 1 mill.	20% ±1%	5% ±2%
Engineering Hours 10,000	2% ±1%	10% ±2.5%
Worst Case B/C ratio	(0+40-30)/(21+3) =**0.42**	(80-20+5)/(7+12.5) = **3.33**
"Worst Worst" case considering estimate credibility factor	0.8 x 0.42= **0.33**	0.2 x 3.33= **0.67**

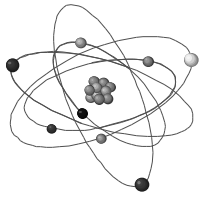
A's Credibility=0.8 (High)

B's Credibility=0.2 (Low)

Slide **62**

10. Results Orientation

- You must keep your focus on the essential results, and never fall victim to the means.
- *"Perfection of means and confusion of ends seem to characterize our age"*
  – *Albert Einstein.*



Software Engineering Productivity Study

- An example of setting objectives for process improvement
- For 1997 Multinational Electronics Company with 70% software labor development content in products
  -

Levels of objectives.

- **1. Fundamental Objectives (above us)**
- **2. Generic Constraints (our given framework)**
  - Political Practical
  - Design Strategy Formulation Constraints
  - Quality of Organization Constraints
  - Cost/Time/Resource Constraints
- **3. Strategic Objectives (objectives at our level)**
- **4. Means Objectives:** (*supporting* our objectives)

Slide **65**

Strategic Objectives

- Support the Fundamental Objectives (Profit, survival)
  - Software Productivity: Lines of Code Generation Ability
  - Lead-Time:
  - Predictability.
  - TTMP: Predictability of Time To Market:
  - Product Attributes:
  - Customer Satisfaction:
  - Profitability:

Slide **66**

Predictability of Time To Market: A sample strategic objective

- **TTMP: Predictability of Time To Market:**
  - » *Gist: From Ideas created to customers can use it. Our ability to meet agreed specified customer and self-determined targets.*
- **Scale: % overrun of actual Project Time compared to planned Project Time**
  - **Project Time: Defined:** time from the date of Toll-Gate 0 passed, or other Defined Start Event, to, the Planned- or Actually- delivered Date of All [Specified Requirements], and any set of agreed requirements.
  - **Specified Requirements: Defined:** written approved Quality requirements for products with respect to Planned levels and qualifiers [when, where, conditions].
    And, other requirements such as function, constraints and costs.
- **Meter: Productivity Project or Process Owner will collect data from all projects, or make estimates and put them in the Productivity Database for reporting this number.**
- **Past [1994, A-package] < 50% to 100%> <- Palli K. guess.**
  [1994, B-package] 80% ?? <- Urban Fagerstedt and Palli K. guess
- **Record [IBM Federal Systems Division, 1976-80] 0%**
  <- RDM 9.0 quoting Harlan Mills in IBM SJ 4-80
- ***"all projects on time and under budget"***
- **[Raytheon Defense Electronics, 1992-5] 0% <- RDE SEI Report 1995 Predictability.**
- **Must [All future projects, from 1999] 5% or less <- discussion level TG**
- **Plan [All future projects, from 1999] 0% or less <- discussion level TG**

Slide **67**

Means Objectives:

- **Support the Strategic Objectives**
  - **Complaints:**
  - **Feature Production:**
  - **Rework Costs:**
  - **Installation Ability:**
  - **Service Costs:**
  - **Training Costs:**
  - **Specification Defectiveness:**
  - **Specification Quality:**
  - **Improvement ROI:**

Slide **68**

Complaints:
a sample 'means' objective

- **Complaints:**
  - *"Customer complaint rate to us"*
- *Gist:*
  - Means Goal: for Customer Satisfaction (Strategic).
- Scale: number of complaints per customer in [defined time into <operation>]
- Past [Syracuse Project , 1997] ?? <bad> <- ML
- Plan [Long term, software component, in first 6 months in Operation] zero complaints <- R PROJECT 96 1.1 b
- "zero complaints on software features"

Slide **69**

Strategies
Intended to impact strategic objectives

- **(means to achieve objectives)**
- **Evo [Product development]:(serious)**
- **DPP [Product Development Process]: Defect Prevention Process.**
- **Inspection?**
- **Motivation.Stress-Management-AOL**
- **Motivation.Carrot**
- **DBS**
- **Automated Code Generation**
- **Requirement -Tracability**
- **Competence Management**
- **Delete-Unnecessary -Documents**
- **Manager Reward:?**
- **Team Ownership:?**
- **Manager Ownership:?**

- **Training:?**
- **Clear Common Objectives:**
- **Application Engineering area:**
- **Brainstormed List (not evaluated or prioritized yet)?**
- **Requirements Engineering:**
- **Brainstormed Suggestions?**
- **Engineering Planning:**
- **Process Best Practices: (silly)**
- **Brainstormed Suggestions?**
- **Push Button Deployment:**
- **Architecture Best Practices:**
- **Stabilization:**
- **World-wide Co-operation?**

A mixture of silly and serious strategies! 2 examples given.

Slide **70**

US Army Example: PERSINSCOM

STRATEGIES ➔ OBJECTIVES	Technology Investment	Business Practices	People	Empow-erment	Principles of IMA Management	Business Process Re-engineering	SUM
Customer Service ?➔0 Violation of agreement	50%	10%	5%	5%	5%	60%	185%
Availability 90% ➔ 99.5% Up time	50%	5%	5-10%	0	0	200%	265%
Usability 200 ➔ 60 Requests by Users	50%	5-10%	5-10%	50%	0	10%	130%
Responsiveness 70% ➔ ECP's on time	50%	10%	90%	25%	5%	50%	180%
Productivity 3:1 Return on Investment	45%	60%	10%	35%	100%	53%	303%
Morale 72 ➔ 60 per mo. Sick Leave	50%	5%	75%	45%	15%	61%	251%
Data Integrity 88% ➔ 97% Data Error %	42%	10%	25%	5%	70%	25%	177%
Technology Adaptability 75% Adapt Technology	5%	30%	5%	60%	0	60%	160%
Requirement Adaptability ? ➔ 2.6% Adapt to Change	80%	20%	60%	75%	20%	5%	260%
Resource Adaptability 2.1M ➔ ? Resource Change	10%	80%	5%	50%	50%	75%	270%
Cost Reduction FADS ➔ 30% Total Funding	50%	40%	10%	40%	50%	50%	240%
SUM IMPACT FOR EACH SOLUTION	***482%***	***280%***	***305%***	***390%***	***315%***	***649%***	
Money % of total budget	15%	4%	3%	4%	6%	4%	
Time % total work months/year	15%	15%	20%	10%	20%	18%	
SUM RESOURCES **BENEFIT/RESOURCES RATIO**	***30*** **16:1**	***19*** **14:7**	***23*** **13:3**	***14*** **27:9**	***26*** **12:1**	***22*** **29:5**	

Sample of Objectives/Strategy definitions

- *Example of one of the Objectives:*

Customer Service:

Gist: Improve customer perception of quality of service provided.

Scale: Violations of Customer Agreement per Month.

Meter: Log of Violations.

Past [1991] Unknown Number ←State of PERSCOM Management Review

Record [NARDAC] 0 ? ← NARDAC Reports 1991

Must : <better than Past, Unknown number> ←CG

Plan [1991, PERSINCOM] 0 "Go for the Record" ← Group SWAG

Technology Investment:

Exploit investment in high return technology.

Impacts: productivity, customer service and conserves resources.

- *An example of one of the strategies defined.*

The summary principle

Motivate people

towards real results

by giving them numeric feedback frequently

and the ability to change anything for success.

SOFTWARE POLICY

- (suggestion, draft)
- Version SEPTEMBER 20 2000
- OWNER:
- Editor: Gilb@acm.org. Detailed practical technical background for this draft see www.result-planning.com and especially 'Priority Management' (116 pages manuscript).

Purpose:

- to define a powerful framework
  - for improving your organization's ability
  - to improve their software organization's capability,
  - as defined in their quantified objectives.
- ***Constraint:***
  - ***this policy should never exceed one physical page, to keep it focussed.***

Slide **75**

.STAKEHOLDER VALUE:

- For all software and systems engineering projects
  - we will formally identify all critical stakeholders, internal and external.
  - We will identify their critical and profitably-served requirements.
  - The requirements will be testable and, if variable,
    - **they will be quantified.**
  - Delivering this defined value to these stakeholders
    - will be the ***primary focus and measure***
    - of all product development process activity.
- ***Rationale: to focus our efforts on critical needs, listen to 'voice of stakeholders'.***

Slide **76**

.ENDS/MEANS CLARITY:

- project requirements will focus on the ***real 'stakeholder-perceived value'*** as the 'requirements'.
  - They will NOT allow design or strategy to replace the real stakeholder needs.
  - Requirements and design to meet those requirements will be ***rigorously separated***
    - in terms of project specification and work processes.
- ***Rationale:***
  - ***extreme clarity of real needs,***
  - ***never confusing this with technology with good intent.***
  - ***Help engineers to focus efforts on serving and competing on the market.***

.NUMERIC CLARITY:

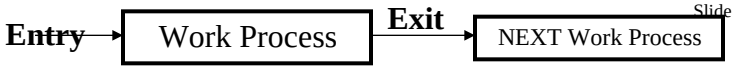
- all notions of qualities (stakeholder values) and costs will
  - in all contexts (requirements, design impacts, project progress, contracts with customers and suppliers)
  - be expressed in terms of **numeric levels on defined scales of measure,**
  - and measured in practice with defined 'Meters'.
  - If it varies, if you can say 'improved',
    - then you must convert these ideas into numbers
    - on defined scales of measure,
    - which become the language of the project.

- ***Rationale:***
  - ***we must have perfect clarity of the stakeholder-critical values,***
  - ***and numeric definition is the ONLY acceptable way to do that.***
  - ***This is necessary for multinational communication.***
  - ***This saves time to market, human resource and will more effectively target our stakeholder values.***
  - ***It allows feedback and correction processes to operate.***

.NUMERIC DEVELOPMENT PROGRESS:

- the primary instrument for ***tracking development progress*** will be
  - the numeric progress for defined stakeholder values (product and service qualities)
    - towards defined and agreed targets,
    - with respect to time.
  - A secondary set of measures will be with respect to the costs or resources planned.

- ***Rationale:***
  - ***this management tracking concept is intended to allow projects to monitor their own progress realistically,***
  - ***using the same measures which any other level of managers would use to judge them.***
  - ***It is intended to be the main component for discussion and evaluation for any meeting, review, milestone or judgement.***
  - ***It should replace conventional milestone progress reporting.***



.WORK PROCESS ENTRY/EXIT CONTROL:

- All software engineering specifications (from contract to code)
  - will be subject to formal entry and exit control.
  - This is primarily numeric and based on 'Major defects remaining' levels,
    - i.e. economic suitability for downstream work processes.
    - Default level maximum 1 major defect remaining per page.
- ***Rationale:***
  - ***to make sure that poor specification practices do not***
    - ***pollute downstream activity,***
    - ***and threaten time to market, human resources or product quality.***

The Ten Most Powerful Principles for Quality in (Software and) Software Organizations

Tom Gilb, Result Planning Limited.
QWE Wednesday 22nd Nov 2000

Summary:
Software knows it has a problem. Solutions abound. But which solutions work? What are the most fundamental underlying principles we can observe in successful projects? This paper presents10 powerful principles that are not widely taught or appreciated. They are based on ideas of measurement, quantification and feedback. Our maturity level with respect to 'numbers' is known to be poor. Hopefully, as we move to higher maturity levels we will also begin to appreciate the power of measurement and numeric expression of idea. What can we do right now? I suggest the first step is to recognize that all your quality requirements can and should be specified numerically. I am not talking about 'counting bugs'. I am talking about quantifying qualities such as security, portability, adaptability, maintainability, robustness, usability, reliability and performance. Decide to make them numeric on your project. Draft some numeric requirements today, surprise your team tomorrow!

Tom Gilb, Result Planning Limited.
www.Result-Planning.com. Email: Gilb@acm.org

Introduction

All projects have some degree of failure, compared to initial plans and promises. Far too many software projects fail totally. In the mid 1990s, the US Department of Defense estimated that about half of their software projects were total failures! (Source N Brown). The civil sector is no better (Morris, 1995). So what can be done to improve project success? This paper outlines ten key principles of successful software development methods, which characterize best practice.
These 10 most powerful software quality principles are selected because there is practical experience showing that they really get us control over qualities, and over the costs of qualities. They have a real track record. This record often spans decades of practice in companies like IBM, HP and Raytheon. There is nothing 'new' about them. They are classic. But the majority of our community is young and experientially new to the game, so my job is to remind us of the things that work well. Your job is to evaluate this information and start getting the improvements your management wants in terms of quality and the time and effort needed to get it.
"Those who do not learn from history, are doomed to repeat it" (Santayana, 1903, The Life of Reason).

Principle 1: Use Feedback

Experience of formal feedback methods is decades old, and many do appreciate their power. However, far too many software engineers and their managers are still practicing low-feedback methods, such as Waterfall project management (also known as Big Bang, Grand Design). Far too many also are checking the quality of their systems by relying on testing, 'late in the day', when they have finished producing their entire system. Even many textbooks and courses continue to present low-feedback methods. This is not from conscious rejection of high-feedback methods, but from ignorance of the many successful and well-documented projects, which have detailed the value of feedback.
Methods using feedback succeed; those without seem to fail. 'Feedback' is the single most powerful principle for software engineering. (Most of the other principles in this paper are really ideas, which support the use of feedback.) Feedback helps you get better control of your project by providing facts about how things are working in practice. Of course, the presumption is that the feedback is early enough to do some good. This is the crux: rapid feedback. We have to have the project time to make use of the feedback (for example, to radically change direction, if that is necessary). Some of the most notable rapid high-feedback methods include:

Defect Prevention Process [originated Mays and Jones, IBM 1983] The Defect Prevention Process (DPP) equates to Software Engineering Institute CMM Level 5 as practiced at IBM from 1983-1985 and on. See (Mays, 1993). DPP is a successful way to remove the root causes of defects. In the short term (a year) about 50% defect reduction can be expected; within 2-3 years, 70% reduction (compared to original level) can be experienced and over a 5-8 year timeframe, 95% defect reduction is possible (Sources: IBM Experience, Raytheon Experience (Dion, 1995)).
The key feedback idea is to 'decentralize' the initial causal analysis activity investigating defects to the grass roots programmers and analysts. This gives you the true causes and acceptable, realistic change suggestions. Deeper 'cause analysis' and 'measured process-correction' work can then be undertaken outside of deadline-driven projects by the more specialized and centralized Process Improvement Teams.
The feedback mechanisms are many. For example, same-day feedback is obtained from the people working with the specification and, early numeric process change-result feedback is obtained from the Process Improvement Teams.

Inspection [originated Fagan, IBM 1975] The Inspection method originated in IBM in work carried out by M. Fagan, H. Mills ('Cleanroom') and R. Radice (CMM inventor). It was originally primarily focussed on bug removal in code and code design documents. Many continue to use it in this way today. However, Inspection has changed character in recent years. Today, it can be used more cost-effectively by focussing on measuring the Major defect level (software standards violations) in sample areas (rather than processing the entire document) of any software or upstream marketing specifications (Gilb, 1993). The defect level measurement should be used to decide whether the entire specification is fit for release (exit) downstream to be used, say for a 'go/no-go' decision-making review or for further refinement (test planning, design, coding).
The main Inspection feedback components are:
• feedback to author from colleagues regarding compliance with software standards.
• feedback to author about required levels of standards compliance in order to consider their work releasable.

Evolutionary Project Management [originated large scale within 'Cleanroom' methods, Mills, IBM 1970] Evolutionary Project Management (Evo) has been successfully used on the most demanding space and military projects since 1970 (Mills, 1980; May 1996; Cotton 1996; Gilb 1988; Gilb 2000). The US Department of Defense changed their software engineering standard (2167a) to an Evo standard (MIL-STD-498, which derived succeeding public standards (for example, IEEE)). The reports (op. cit.) and my own experience, is that Evo results in a remarkable ability to delivery on time and to budget, or better, compared to conventional project management methods (Morris, 1994).
An Evo project is consciously divided up into small, early and frequently delivered, stakeholder result-focussed steps. Each step delivers benefit and build towards satisfaction of final requirements. Step size is typically weekly or 2% of total time or budget. This results in excellent regular and realistic feedback about the team's ability to deliver meaningful measurable results to selected stakeholders. The feedback includes information on design suitability, stakeholders' reaction, requirements' tradeoffs, cost estimation, time estimation, people resource estimation, and development process aspects.

Statistical Process Control [originated Shewhart, Deming, Juran: from 1920's] Statistical Process Control (SPC) although widely used in manufacturing (Deming, 1986) is only to a limited degree actually used in software work. Some use is found in advanced Inspections (Dion, 1995; Florac, 1997). The Plan Do Study (or Check) Act cycle is widely appreciated as a fundamental feedback mechanism.

Principle 2: Identify Critical Measures
It is true of any system, that there are several factors, which can cause a system to die. It is true of your body, your organization, your project and your software or service product. Managers call them 'Critical Success Factors.' If you analyzed systems looking for all the critical factors, which caused shortfalls or failures, you would get a list of factors needing better control. They would include both stakeholder values (such as serviceability, reliability, adaptability, portability and usability) and the critical resources needed to deliver those values (such as people, time, money and data quality).You would find, for each of these critical factors, a series of faults, which would include:

• failure to systematically identify all critical stakeholders and their critical needs
• failure to define the factor measurably. Typically, only buzzwords are used and no indication is given of the survival failure) and target (success) measures
• failure to define a practical way to measure the factor
• failure to contract measurably for the critical factor
• failure to design towards reaching the factor's critical levels
• failure to make the entire project team aware of the numeric levels needed for the critical factors
• failure to maintain critical levels of performance during peak loads or on system growth.

Our entire culture and literature of 'software requirements' systematically fails to account for the majority of critical factors. Usually, only a handful, such as performance, financial budget and deadline dates are specified. Most quality factors are not defined quantitatively at all. In practice, all critical measures should always be defined with a useful scale of measure. However, people are not trained to do this, and managers are no exception. The result is that our ability to define critical 'breakdown' levels of performance and to manage successful delivery is destroyed from the outset.

Principle 3: Control Multiple Objectives

You do not have the luxury of managing qualities and costs at whim. You cannot decide for a software project to manage just a few of the critical factors, and avoid dealing with the others. You have to deal with *all* the potential threats to your project, organization or system. You must simultaneously track and manage all the critical factors. If not, then the 'forgotten factors' will probably be the very reasons for project or system failure.

Table 1: An example of an IE table.

	Step #1 Plan A: {Design-X, Function-Y}	Step #1 Actual	Difference. -is bad + is good	Total Step 1	Step #2 Plan B: {Design Z, Design F}	Step #2 Actual	Step #2 Difference	Total Step 1+2	Step #3 Next step plan
Reliability 99%-99.9%	50% ±50%	40%	-10%	40%	30% ±20%	20%	-10%	60%	0%
Performance 11 sec.-1 sec.	80% ±40%	40%	-40	40	30% ±50%	30%	0	70%	30%
Usability 30 min. -30 sec.	10% ±20%	12%	+2%	12%	20% ±15%	5%	-15%	17%	83%
Capital Cost 1 mill.	20% ±1%	10%	+10%	10%	5% ±2%	10%	-5%	20%	5%
Engineering Hours 10,000	2% ±1%	4%	-2%	4%	10% ±2.5%	3%	+7%	7%	5%
Calendar Time	1 week	2 weeks	-1 week	2 weeks	1 week	0.5 weeks	+0.5 wk	2.5 weeks	1 week

I have developed the Impact Estimation (IE) method to enable tracking of critical factors, but it does rely on critical objectives and quantitative goals having been identified and specified. Given that most software engineers have not yet learned to specify *all* their critical factors *quantitatively* (Principle 2), this *next* step, tracking progress against quantitative goals (this principle), is usually impossible.

IE is conceptually similar to Quality Function Deployment (Akao, 1990), but it is much more objective and numeric. It gives a picture of reality that can be monitored (Gilb, 1988, Gilb, 2000). See Table 1, an example of an IE table. It is beyond the scope of this paper to provide all the underlying detail for IE. To give a brief outline, the percentage (%) estimates (see Table 1) are based, as far as possible, on source-quoted, credibility evaluated, objective documented evidence. IE can be used to evaluate ideas *before* their application, and it can also be used (as in Table 1) to track progress towards multiple objectives *during* an Evolutionary project. In Table 1, the 'Actual' and 'Difference" and 'Total' numbers represent *feedback* in small steps for the chosen set

of critical factors that management has decided to monitor. If the project is deviating from plans, this will be easily visible and can be corrected on the next step.

Principle 4: Evolve in small steps

Software engineering is by nature playing with the unknown. If we already had exactly what we needed, we would re-use it. When we choose to develop software, there are many types of risk, which threaten the result. One way to deal with this is to tackle development in small steps, one step at a time. If something goes wrong, we will immediately know it. We will also have the ability to retreat to the previous step, a level of satisfactory quality, until we understand how to progress again.

It is important to note that the small steps are not mere development increments. The point is that they are incremental satisfaction of identified stakeholder requirements. Early stakeholders might be salespeople needing a working system to demonstrate, system installers/help desk/service/testers who need to work with something, and finally, early trial users.

The duration of each small step is typically a week or so. The smallest widely reported steps are the daily builds used at Microsoft, which are useful-quality systems. They cumulate to 6-10 week 'shippable quality' milestones (Cusomano, 1995).

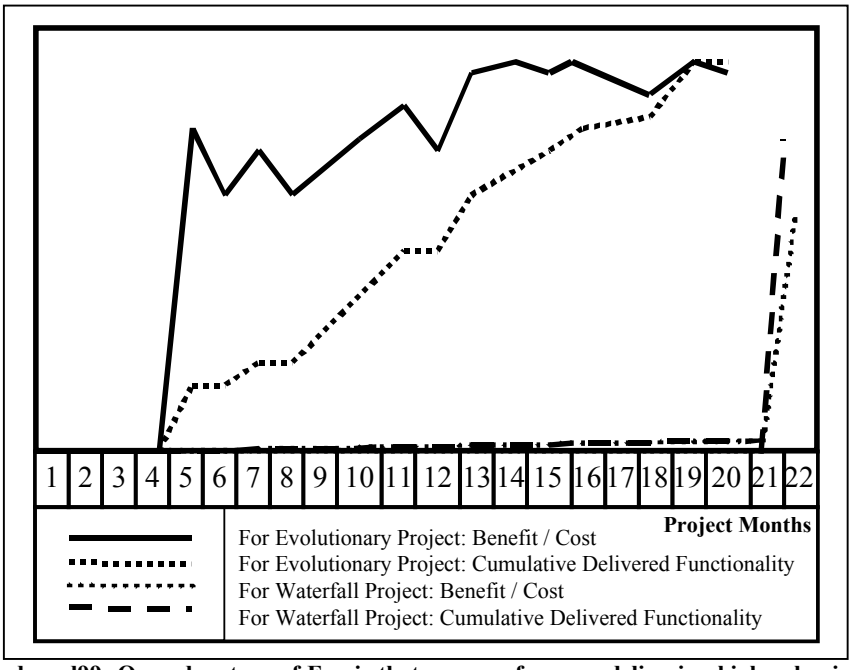


Figure 1: From Woodward99: One advantage of Evo is that you can focus on delivering high value increments to critical stakeholders early. The upper line represents high value at early stages.

Principle 5: A stitch in time saves nine

Quality Control must be done as early as possible, from the earliest planning stages, to reduce the delays caused by finding defects later. There needs to be strong specification standards (such as 'all quality requirements must be quantified') and rigorous checking to measure that the rules are applied in practice. When the specifications are not of some minimum standard (like < 1 Major defect/page remaining) then they must be edited until they become acceptable.

- Use Inspection sampling to keep costs down, and to permit early, before specification completion, correction and learning.
- Use numeric Exit from development processes, such as "Maximum 0.2 Majors per page".

It is important that quality control by Inspection be done very early for large specifications, for example within the first 10 pages of work. If the work is not up to standard then the process can be corrected before more effort is wasted. I have seen half a day of Inspection (based on a random sample of 3 pages) show that there were about 19 logic defects per page in 40,000 pages of air traffic control logic design (1986, Sweden). The same

managers, who had originally 'approved' the logic design for coding carried out the Inspection with my help. Needless to say the project was seriously late.
In another case I facilitated (USA, 1999, Jet parts supplier) eight managers sampled two pages out of an 82 page requirements' document and measured that there were 150 'Major' defects per page. Unfortunately they had failed to do such sampling three years earlier when their project started, so they had already experienced one year of delay; and told me they expected another year delay while removing the injected defects from the project. This two-year delay was accurately predictable given the defect density they found, and the known average cost from Major defects. They were amazed at this insight, but agreed with the facts. In theory, they could have saved two project years by doing early quality control against simple standards: clarity, unambiguous and no design in requirements were the only rules we used.
These are not unusual cases. I find them consistently all over the world. Management frequently allows extremely weak specifications to go unchecked into costly project processes. They are obviously not managing properly.

Principle 6: Motivation moves mountains
Motivation is everything! When individuals and groups are not motivated positively. They will not move forward. When they are negatively motivated (fear, distrust, suspicious) they will resist change to new and better methods. Motivation is a type of method. In fact there are a lot of large and small items contributing to your group's 'sum of motivation'. We can usefully divide the 'motivation problem' into four categories:

- the will to change
- the knowledge of change direction
- the ability to change
- the feedback about progress in the desired change direction.

Leaders (I did not say 'managers') create the will to change by giving people a positive, fun, challenge and, the freedom and resources to succeed. John Young, CEO of Hewlett Packard during the 1980's, inspired his troops by saying that he thought they needed to aim to be measurably ten times better in service and product qualities ("10X") by the end of the decade (1980-1989). He did not demand it. He supported them in doing it. They failed. Slightly! They reported getting about 9.95 times better, on average, in the decade. The company was healthy and competitive during a terrible time for many others, such as IBM.
The knowledge of change direction is critical to motivation; people need to channel their energies in the right direction! In the software and systems world, this problem has three elements, two of which have been discussed in earlier principles:
• measurable, quantified clarity of the requirements and objectives of the various stakeholders (Principle 2)
• knowledge of all the multiple critical goals (Principle 3)
• formal awareness of constraints, such as resources and laws.
These elements are a constant communication problem, because:
• we do not systematically convert our 'change directions' into crystal clear measurable ideas; people are unclear about the goals and there is no ability to obtain numeric feedback about movement in the 'right' direction. We are likely to say we need a 'robust' or 'secure' system; and less likely to convert these rough ideals into concrete, measurable, defined, agreed, requirements or objectives.
• we focus too often on a single measurable factor (such as '% built' or 'budget spent') when our reality demands that we simultaneously track multiple critical factors to avoid failure and to ensure success. We don't understand what we should be tracking, and we don't get enough 'rich' feedback.

Principle 7: Competition is eternal
Our conventional project management ideas strongly suggest that projects have a clear beginning and a clear end. In our competitive world, this is not as wise a philosophy as the one Deming suggests, "Eternal Process improvement is necessary as long as you are in competition." We can have an infinite set of 'milestones' or evolutionary steps of result delivery, and use them as we need; the moment we abandon a project, we hand opportunity to our competitors. They can sail past our levels of performance, and take our markets.
The practical consequence is that our entire mindset must always be on setting new ambitious numeric 'stakeholder value' targets, both for our organizational capability and for our product and service capabilities.

Continuous improvement efforts in the software and services area at IBM, Raytheon and others (Mays, 1993; Dion, 1995; Kaplan, 1994; Hewlett Packard (10X, Young)) show that we can improve critical cost and performance factors by 20 to 1, in five- to eight-year timeframes. Projects must become *eternal* campaigns to get ahead and stay ahead.

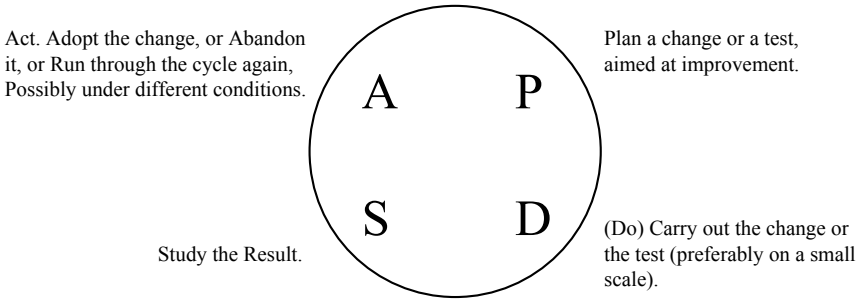


Figure 2: The Shewhart Cycle for Learning and Improvement - the P D S A Cycle.
Reproduction from a letter from W. Edwards Deming, 18 May, 1991 to the author.

Principle 8: Things take time

"It takes 2-3 years to change a project, and a generation to change a culture." W. Edwards Deming
"Things Take Time" (TTT). Piet Hein (Denmark)

Technical management needs to have a long-term plan for improvement of the critical characteristics of their organization and their products. Such long-term plans need to be numerically trackable, and to be stated in multiple critical dimensions. At the same time visible short-term progress towards those long-term goals should be planned, expected and tracked.

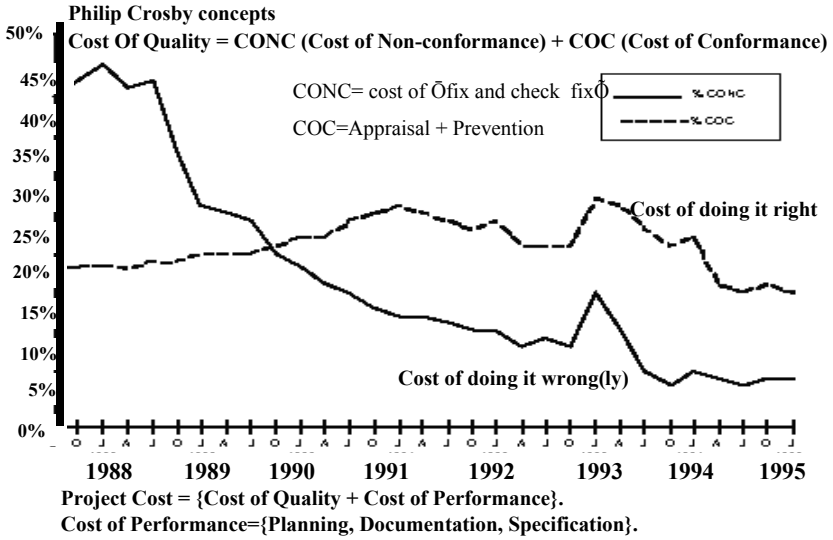


Figure 3: Cost of Quality versus Time: Raytheon 95 - the 8 year evolution of rework reduction. In the case of Raytheon process improvements (Dion, 1995), many years of persistent process change for 1,000 programmers was necessary to drop rework costs from 43% of total software development costs, to below 5%.

Principle 9: The bad with the good

Any method (means, solution, design) you choose will have multiple quality and cost impacts, whether you like them or not! In order to get a correct picture of how good any idea is, for meeting our purposes, we must

• have a quantified multidimensional specification of our requirements; our quality objectives and our resources (people, time, money)
• have knowledge of the expected impact of each design idea on all these quality objectives and resources
• evaluate each design idea with respect to its total, expected or real, impact on our requirements; the unmet objectives and the unused cost budgets.

We need to estimate all impacts on our objectives. We need to reduce, avoid or accept negative impacts.
We must avoid simplistic one-dimensional arguments. If we fail to use this systems engineering discipline, then we will be met with the unpleasant surprises of delays, and bad quality, which seem to be the norm in software engineering today. One practical way to model these impacts is using an IE table (as in Table 1).

Principle 10: Keep your eye on where you are going

"Perfection of means and confusion of ends seem to characterize our age" Albert Einstein

To discover the 'real' problem we have only to ask of a specification: "Why?" The answer will be a higher level of specification, nearer the real ends. There are too many designs in our requirements!
You might say, why bother? Isn't the whole point of software to get the code written? Who needs high level abstractions; cut the code! But somehow that code is late and of unsatisfactory quality. The reason is often lack of attention to the real needs of the stakeholders and the project. We need these high-level abstractions of what our stakeholders need, so that we can focus on giving them what they are paying us for! Our task is to design and deliver the best technology to satisfy their needs at a competitive cost.
One day, software engineers will realize that the primary task is to satisfy their stakeholders. They will learn to design towards stakeholder requirements (multiple simultaneous requirements!). One day we will become real systems engineers, and we realize that there is far more to software engineering than writing code!

Conclusion

Motivate people, towards real results, by giving them numeric feedback frequently and the freedom to use any solution, which gives those results. It is that simple to specify. It is that difficult to do.

References

Yoji Akao (Editor): Quality Function Deployment: Integrating Customer Requirements into Product Design. Productivity Press, Cambridge Mass. USA, 1990.
Todd Cotton: Evolutionary Fusion: A Customer-Oriented Incremental Life Cycle for Fusion. Hewlett-Packard Journal, August 1996, Vol. 47, No. 4, pages 25-38.
Michael A. Cusumano and Richard W. Selby: Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. The Free Press (a division of Simon and Schuster), 1995, ISBN 0-028-74048-3, 512 pages.
W. Edwards Deming: Out of the Crisis. MIT CAES Center for Advanced Engineering Study, Cambridge MA USA-02139, 1986, ISBN 0-911-37901-0, 507 pages, hard cover.
Raymond Dion: The Raytheon Report. http://www.sei.cmu.edu/products/publications/95.reports/95.tr.017.html.
Michael E. Fagan: Design and code inspections. IBM Systems Journal, Vol. 15 (3), pages 182-211, 1976. Reprinted IBM Systems Journal, Vol. 38, Nos. 2&3, 1999, pages 259-287. www.almaden.ibm.com/journal
Tom Gilb: Principles of Software Engineering Management. Addison-Wesley, UK/USA, 1988, 442 pages. ISBN 0-201-19246-2, Soft-cover, £24.95.
Tom Gilb and Dorothy Graham: Software Inspection. Addison-Wesley, 1993 471 pages. ISBN 0-201-63181-4. Japanese Translation, August 1999, Yen 5,000. 450 pages including index, Soft-cover. ISBN 4-320-09727-0, C3041 (code next to ISBN).
Tom Gilb: Competitive Engineering, Addison-Wesley, UK, End 2000.
There will always be a version of this book free at my website, http://www.result-planning.com
Craig Kaplan, Ralph Clark and Victor Tang: Secrets of Software Quality, 40 Innovations from IBM. McGraw Hill, ISBN 0-079-11975-3, 383 pages.
Ralph L. Keeney: Value-focused Thinking: A Path to Creative Decision-making. Harvard University Press, Cambridge Mass/London, 1992, ISBN 0-674-93197-1.
Elaine L. May and Barbara A. Zimmer: The Evolutionary Development Model for Software. Hewlett-Packard Journal, August 1996, Vol. 47, No. 4, pages 39-45.
Robert Mays: Practical Aspects of the Defect Prevention Process. (Gilb, 1993, Chapter 17 written by Mays)
Harlan D. Mills: IBM Systems Journal, 1980 (4). Also republished IBM Systems Journal, Nos. 2&3, 1999.
Peter W. G. Morris: The Management of Projects. Thomas Telford, London, 1994, ISBN 0-727-7169-3.
Tom Peters: Reinventing Work, The Project 50. Alfred A. Knopf, New York, 2000, ISBN 0-375-40773-1.
William A. Florac, Robert E. Park and Anita D. Carleton: Practical Software Measurement: Measuring for Process Management and Improvement. Guidebook from Software Engineering Institute, Reference: CMU/SEI-97-HB-003, Downloadable from SEI web site (Acrobat Reader): ftp://ftp.sei.cmu.edu/ for publications, and main site http://www.sei.cmu.edu, 1997, 240 pages.
Stuart Woodward: Evolutionary Project Management. IEEE Computer, Oct 1999, pages 49-57.

QWE2000 Keynote Session K1-2

Mr. Jens Pas
(I2B)

Test Out-Sourcing: From Necessary Evil to E-Competitive Advantage

Key Points

- Test Outsourcing as a strategic organisation model
- Humanity as a cause of errors
- Human behaviour when dealing with errors
- The benefits of a dedicated Test Partner

Presentation Abstract

This presentation explains why Test Outsourcing is the only way to organise testing correctly. Test Outsourcing is explained as a strategic and corporate decision. It concerns setting up long-term relationships with test services suppliers. The motivation behind this idea is found in the paper below.

The presentation starts by explaining from a human perspective how errors are made. It explains why our humanity is the most real cause of bugs. In a second part the presentation discusses our human behaviour when encountering errors. Here the logic is built to come to the solution of Test Outsourcing. The third chapter of the presentation provides first a model on how to define responsibilities and secondly gives an example of how developers and testers should be organised in an outsourced situation. Examples are given of the distribution of tasks, the use of metrics and the definition of the accountabilities.

Where relevant, real cases will be used to illustrate the model. These real cases are not mentioned in the paper below for reasons of confidentiality.

About the Speaker

Jens Pas is Managing and founding Partner of I2B and President of the Board of Directors. Jens Pas is an authority in the business of Software Testing. He developed a unique method for structured software testing. His method was introduced in and adopted by major Belgian and Dutch companies. He has also development Test Assessment programs and has conducted them at directors and top-management level at various sites in larger IT environments. He published various articles regarding the methodology of Software Testing and is the author of the textbook that accompanies the course ôIntroduction to Structured Software

Testingö. He is a speaker and track chairman on many Software Engineering and Quality Conferences (Eurostar 98 Munich, 99 Barcelona & 2000 Copenhagen, Quality Week Europe, Fesma Brussels, Software Automation, Novi Amsterdam, Technological Institute of the Royal Society of Belgian Civil Engineers, SQE Europe Test Congress 2000). Currently he is specialising in Innovation Management services.

Jens Pas was co-founder and General Manager of a Belgian consultancy company specialised in testing. Previously he was responsible for the re-engineering of the pan-European IT-support department of Quaker Oats Europe and for setting up the Application Support department of the ETS division of Barco Graphics.

Jens Pas is an engineer and holds an Executive International Master in Business Administration degree from the Vlerick Leuven Gent Management School.

I2B

TEST OUTSOURCING

FROM NECESSARY EVIL TO COMPETITIVE ADVANTAGE

By

Jens Pas
Managing Director – CEO
Idea to Business (I2B)
Kwaadstraat 25
B-9750 Zingem
Belgium
Tel.: +32 (9) 384.86.27
Fax: +32 (9) 384.30.46
E-mail: jens.pas@i2b.be
http://www.i2b.be

ABSTRACT

This presentation explains why Test Outsourcing is the only way to organise testing correctly. Test Outsourcing is explained as a strategic and corporate decision. It concerns setting up long-term relationships with test services suppliers. The motivation behind this idea is found in the paper below.

The presentation starts by explaining from a human perspective how errors are made. It explains why our humanity is the most real cause of bugs. In a second part the presentation discusses our human behaviour when encountering errors. Here the logic is built to come to the solution of Test Outsourcing. The third chapter of the presentation provides first a model on how to define responsibilities and secondly gives an example of how developers and testers should be organised in an outsourced situation. Examples are given of the distribution of tasks, the use of metrics and the definition of the accountabilities.

Who gets hired as a programmer when he admits during the interview that he will program errors? Who gets hired as a programmer when he claims that he only wants to be hired if his friend, the tester, gets hired as well? No one. Which manager states that the software his company makes is free of errors? Also no one. What is it that makes people believe that they are doing a good job whilst at the same time they know with empirical evidence that their good job isn't as good as it is intended to be…

The above paradox is the fundamental cause why still today testing has hard times getting introduced in organisations. Even though we all know and by now even agree that testing is an essential activity of our software engineering process, getting it organised remains a significant weakness.

It is a fact. We make errors, certainly when we try to make software. We all have heard the issues of software being intangible and complex, causing us to confuse things and forget links between the different parts of code that we've written. Hence, we make a lot of errors. For many years, solutions to increase the quality of software has been sought in improvements in programming techniques and setting up software "engineering" processes. We tried to develop many preventative solutions to reduce the amount of bugs made. And when we still concluded that too many defects slipped through, we installed remedial activities such as system tests, factory acceptance tests, site acceptance tests, parallel installations, beta-installations and pilot sites. Each and every one of these activities costs a lot of money and still today they have not given a satisfactory result. Automatic cash-distributors (ATM's) still get blocked on a busy Saturday afternoon and pc's still crash at moments we did not save our documents. I even experienced it when writing this paper. So, there is indeed still a lot to be done to solve the problem of bad software quality.

This paper will not discuss the preventative and remedial actions that can be taken to improve software quality. The paper will not even discuss which software testing methods that must be applied. The reader can find these elements covered in the many presentations of the Quality Week conference and in the increasingly growing literature on software testing.

This paper will look at testing from a human behavioural point of view. It will develop a logic that will lead to testing solutions that will find their implementation through the organisation of these humans. In a first part, we will discuss the cause of error creation from a human perspective. In a second part we will analyse the human behaviour when facing these failures. The final part of the document will describe how we should tackle the error prevention, creation, detection and repairing process with a sound test organisation.

WHY WE MAKE ERRORS

We have empirical evidence that we do make errors. This is not an axiom. It is a fact. Many studies have been conducted to find the root causes of these errors. Communication has been pointed at as being the most popular cause of errors. In short, users are not able to completely communicate to the programmers what they need. And vice versa, programmers are not able to fully understand what users are explaining. Hence a lot of misunderstandings, assumptions and consequently bad software are being made. Our experience in computer science has also been seen for a long time as one of the key problems. The fact that so-called software engineering processes are more craftsmen-activities has proven to be responsible for an irregular pattern of software quality levels. Our experience has, however increased and many initiatives have been taken to improve the level of engineering in our software creation process. CMM, Spice, Tick-IT are examples of the latter.

One very difficult cause of errors to measure or investigate is we, humans. It was Dorothy Graham that claimed in the beginning of the nineties that humanity itself was a major source of problems. "**Humans and computers are incompatible**", she said. Graham explained that humans have four characteristics that make them make errors. For one, humans are very limited when it comes to managing complexity. Try for instance to memorise the following 14-digit number:

13655212283031

We will use this number later…

If you are a programmer, you experienced the difficulty to understand your own lines of code, even of a straightforward procedure, when you haven't had a look at it for a couple of months. It is a fact, we cannot grasp complex information. In marketing this human limitation is known. There is a marketing law that states that commercial one-liners should not exceed more then seven words. Above seven, people forget some of them. It is not the purpose of this paper to explain why we cannot manage a high level complexity but it is somewhere related to the fact that we interpret things rather than register. Those people who have seen the movie Rainman, where Dustin Hoffman plays Raymond Babbit, an autistic man, have seen that autistic people can manage complexity much easier than "normal" people can. Autistic people have a lower level of interpretation. They take things rather literally. Circumstantial information does not distract them, which in most cases is essential to understand the information received. Raymond Babbit can count all the matches that fell out of a matchbox and that are scattered around on the ground with the blink of an eye. When we look at the falling sticks, we see chaos, we start thinking of how to clean it up and we lose focus on the real fact. Raymond Babbit stops in the middle of the street when he notices that the pedestrian sign turns from "Walk" into "Don't Walk". We interpret the circumstances and we know that the "Don't Walk" sign only means don't walk if we are still on the sidewalk of the street. "Don't Walk" even means "walk faster" when we are in the middle of the street, the opposite. Understanding complexity is one of our weaknesses.

Second, Graham states that humans are optimistic. We are optimistic, but so what? Being optimistic means that we believe that what we've done was OK, without errors. We have strong conviction that our work was OK. If we would've made one or two mistakes we would have noticed and repaired it immediately. Our optimism regarding our results creates blind spots. We are not able to see our own errors.

This optimism ties into our subconsciousness that rules over our behaviour. Everyone has the experience that when proof-reading the letter one write himself reveals less errors then having the letter read by another person. You, the writer, know what you mean with your text and as such assume that what is written is clear and understandable. Words that are missing and typos are overlooked as a consequence of our involvement with the text.

Finally, humans are creative. We invent things as we go along. How well structured our plans and designs might be, once we start programming we add and change things, sometimes even without noticing (again this subconsciousness). Typical is the situation where we discover a new feature in our programming environment that simplifies (so it looks at first) a way of coding a procedure. We are convinced that this new feature can substitute the other way we originally designed our algorithm and we start writing code that differs from the design. Having narrowed down our focus to this local piece of code, we forget the impact the change has on other parts of the software. We discover much later that the use of the new feature has caused a problem in a part of the software lines away from the place where we altered the design. We discover this problem most probable when doing system or acceptance tests. Since the design does not match the code anymore, we can hardly trace back the root of the problem. This is one very typical example of the consequences of our creativity. There are others. It has been proven that some software, particularly in banks, where programs have a large legacy and a strong integration, can

add up to 80% of dead code. Dead code is code which is not used anymore but which still is part of the program. Nobody dares to remove the code since nobody knows what the impact will be…

In short, humans are limited, optimistic, use their undocumented subconsciousness and cannot help being creative and as such create bugs.

Do you still remember the number you memorised on the previous page? If not, remember that 1 normal calendar year counts 365 days, 52 weeks and 12 months which have 28, 30 or 31 days. You will not forget this 14-digit number anymore…You see, we need structure and interpretation…

OUR ATTITUDE AGAINST ERRORS

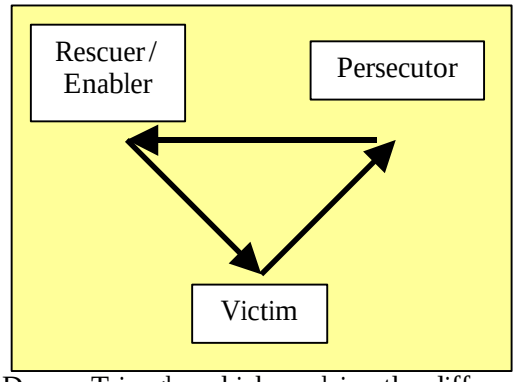
We now know that our human behaviour is responsible for the creation of errors. If we know all this, how come that we don't prevent this from happening. If we understand the mechanism of the human subconsciousness, why can't we prevent it from interfering?

By the nature of the cause, humanity, we cannot prevent making errors. If we would be able to eliminate that we would have to eliminate humanity. We can reduce human intervention – this is the field of software engineering – but we cannot eliminate humanity itself. Even if we could, we would eliminate also the source that is capable of creating things. Humanity might be a cause of errors; it is also the element that allows us to make software.

So, we must catch the errors before they can cause any harm. Here, a new human constraint surfaces. We know that we are creative creatures, assuming that we make qualitative things. We organise ourselves in organisations and companies to work together and realise great products. We organise ourselves in constructive teams that strive towards the creation of sound products. Our complete way of interaction with each other is based on this model of constructive teams. We define incentives related to the creation of good products. We measure our importance and status based on the "constructive part" that we cover in the product creation process. Software programmers consider themselves much more as the father or even owner of the program they wrote then the system manager that maintains and manages the development environment. Our complete way of managing organisations is based upon this "value-added" feeling. Companies consider their programmers as their "human capital". Although having a clean desk and office is essential as well, the cleaning personnel is not considered as key element to the creation process of our software. It is here that a new problem lies. The testers, where do they fit in?

Do tester build things? Yes, they make tests. But do they make the products, which the company they work for sells? No, they do not make the programs, nor do they add elements to the program. Many say that the testers improve the quality of the software and as such create added value. This is wrong. Testers do not add any value at all. If we hadn't made the mistakes in the first place, we wouldn't have needed the testers. Testers are a necessary evil. They are a cost, which we preferably would not want to make. Of course, since we know that we cannot prevent errors from appearing we might say that testing is as natural or existential as breathing. Still testing is not creating a product. Testing is verifying and validating what has been created.

Testers don't even fix the bugs. They are not allowed to. It would be very inefficient if testers where allowed to fix bugs. For one, they don't know the code (they should not know) and second they didn't make the errors in the first place. Allowing testers to fix bugs or even making testers responsible for the final quality is a suicide of the testers. Lee Copeland gave a very impressive talk about this at Eurostar 98 in Munich. He explained to all of us what co-dependency was. Co-dependency is a psychological phenomenon where one person takes over the responsibility of someone else's actions. It would lead too far to explain co-dependency in this paper but simply consider how a tester would behave if he were responsible for the final quality of the software. Would he report many defects? Wouldn't the programmers who made all the errors that caused him to be blamed frustrate him? There is a complete psychological model, called Karpmann's Drama Triangle, which explains the different phases a co-dependent person goes through. If a tester were to be responsible for the quality, he would first act as a Rescuer, finding (and repairing) bugs and saving the world. Programmers, who would not be held accountable for the bugs, would not know what has happened and would continue in making the same amount of errors. The tester would start to feel unappreciated since he takes all the blame and nothing improves. The tester would start feeling as being a victim of the situation and of the programmers, which by then he would consider as aggressors. Finally, the tester would start persecuting the programmers. Any feedback that goes back to the programmer in the persecuting atmosphere would not at all be constructive. The tester is by now even the enabler of the bad behaviour of the programmers. Which brings us back to the starting point.



For testers to do their job correct, there is only one thing they should do: **Find bugs!** No matter what nice definitions that are given to the nature of testing: "the effort to check whether programs do what they are supposed to do" or "to check that they don't do what they are not supposed to do". Other combinations of the above statements are also used. This is all worthless. Testers should find errors. That's all there is to it. Nothing more, nothing less. Of course, the objective of the testing activity is to get to a certain point of validation that shows that the software is ok, but the testing work itself is all about finding errors.

If testers would concentrate on trying to prove that the software is ok, they would adapt their behaviour and develop and execute those tests that serve that purpose. This is what people do. Myers already wrote about this behaviour in his book "The Art of Software Testing[1]". People act according to the goal they want to achieve. And they try to find the easiest way to get to it. So, if we stress on the fact that testing is a constructive activity, that it must help us to improve quality, we are very likely to reach that goal as easy as possible, by not finding many bugs. The smaller the amount of bugs found, the faster we can claim that the software is of good quality. Whether this is reality, is a different issue. We do not live by what is real, but by what we perceive as being real.

The problem becomes even worse if we look at the sociological behaviour of our testers. Testers work in an organisation. This organisation and all its employees and co-workers with it are proud of what they realise: a sound competitive product. Organisations praise those who make these sound products. We all know that we await a brilliant career if we prove to be good programmers, analysts or managers. A genius programmer might be awkward to work with, he will still be regarded as key to the corporation. Companies even insure themselves against the risk of losing these key people.

[1] Myers, Glenford, "*The Art of Software Testing*", John Wiley & Sons, 1979

What happens to the genius tester? The guy or girl that is so brilliant in developing nasty tests that reveal the many mistakes our company is making and illustrates how lousy our products really are? Do we praise those people? Or do we call them over-acting critics? Will a tester get a pat on the back when he was capable of finding that bug that prevented the release of the software on the next trade show? Will he get a raise because he "saved the company from trouble which has not happened yet" (=read bad press due to introducing a low quality product into the market). Will the company be spending a large investment budget on testing tools so that we can show even more easily how bad the software is we make? Or will we spend this money on the latest multimedia object library that will make our products "look" good?

Testers are not popular. Testers are a pain in the ass. Testers are a necessary evil. We hate them, we despise them, we want them out of our lives. For all clarity, I'm a tester too. I know what I'm talking about.

HOW WE SHOULD ORGANISE TESTING

As we can see above, it is not possible for a company to genuinely organise their testing properly. Even if we send some of our people to testing courses, our testers will not have the right attitude for finding bugs. Even if we set-up a separate test department for our testers and give them incentives for the amount of bugs they find, we will never do a good job. The internal power of "being the best company with the best products" is so strong that it beats all other internal behaviour that tries to obstruct reaching this goal.

This does not mean that companies are not interested in testing, or that they do not want to invest in testing. In this era of ICT as it is called today (Information & Communication Technology), the need for testing is more then ever present. Computer crashes, stalled programs or other devices that won't do what they are supposed to do continuously hinder us. We even start to understand the costs that are associated with the bugs. A cost, by the way, which is increasing exponentially. The more we become dependent of computers the more severe the impact of the bugs. Companies are interested in testing. This is the good news for all of us testers. We do have a nice and bright future ahead of us. There will be more job openings for testers and we will get our respected place in society. This place however is, not within our current product building organisation.

Testers must be independent from the organisations they are working for. As long as testers remain part of the same company as the programmers, they will always be the underdogs. If it comes to a final decision to release or not to release a product, companies will (almost) never follow the advice of the testers. Consider the introduction of a new television set by a European consumer electronics company. If stalling the release of the TV because of software problems, means that the Japanese competitor will come to the market first, we will most probably take the risk and release an unstable product. The marketing people will not only have the (biased) economical calculations on their side (the fact that a loss of market share is more costly then some "problems" with the first customers), they will also have the political power and intrinsic belief of how good we are on their side.

The only good solution to organise the testing, is to involve external testers, testers from an independent company, a test services supplier. These external testers are not hired bodies you take in, because you didn't have the people or because no one did want the lousy job. We are not talking about hiring capacity here. This is about setting up a business model with one or more external companies that consistently provide the testing services to a particular client. These external companies are partners. They are integrated in your organisation, but they remain external. What is the difference with own testers? The difference lies in the fact that hired testers are considered as experts, experts that are supposed to be good at testing, hence at finding errors.

These external experts have no or very little political pressure. Their supplier-contract manages this pressure. Their lives are all about finding as many defects as they can. That's what they are paid for. The mission statement of professional testing companies is all about helping their clients in achieving high quality by seeking for bugs. The goal of every testing company is to find bugs, not to make qualitative software programs. Professional testing companies are bug busters. That's their reason of existence.

The above-described organisation is not new. It is the classical outsourcing concept. We already discovered that window cleaning, office cleaning and machine maintenance is much better dealt with if we outsource it. Our own cleaning lady was until recently considered as a cost. Consequently we paid here a low wage and still expected her to make everything nice and neat. Preferably without too much of our interventions. We even ask her to come by late at night, when everyone has gone home. The cleaning lady has little pride in her job. She felt as being a necessary evil. One day our cleaning lady leaves our company and become the hygienical operator of the company Clean corp. The mission statement of Clean corp. is *to provide every customer a nice tidy working space*. The hygienical operators (read cleaning ladies) even get skilled in how to clean desks and computer equipment. They are trained in the purposes of certain chemical products; which ones you can mix and which ones you cannot mix. Since we lost our cleaning lady, we hire Clean corp. to do the job and all of a sudden, our cleaning lady is back. But now, she is an expert, walking around with a professional trolley with outstanding cleaning products and a shiny Clean corp. uniform. She even performs preventative checks and looks for optimisation of the waste bin collecting system. When we change our office furniture we even ask her opinion regarding floors carpeting; which ones are easy to maintain and which ones are not.

Through outsourcing, a cost becomes an opportunity. Maintenance companies, who maintain factory equipment, execute preventative maintenance, reducing the downtime of our factory. They give us a new advantage.

Even more than cleaning and maintenance is testing an activity that can genuinely be outsourced. We outsource cleaning and maintenance because they are non-core activities. They are activities that are not considered as our distinctive competence. The outsourcing of testing is different. Testing is a part of the software engineering process. Consequently, it is be part of many companies' competitive advantage. This is why many companies believe that they should master the testing themselves; because it is a part of their core-competence. Testing must be outsourced for human reasons explained above, but not because of the peripheral (non-core) value. The outsourcing of testing is a strategic decision that creates a competitive advantage.

When we decide to outsource testing we must consider the outsourcing structure as setting up a strategic relationship with one or more suppliers. I would advise to have more than one supplier for the testing services, purely for risk-reasons. Single sourcing always creates an unbalance in the relationship between customer and supplier. But that is a different story. The outsourcing of testing means that an external company will get a great deal of insight in the engineering processes of your company, not to speak about the knowledge of the quality of the products. Working with an external testing partner requires a great deal of trust. It is therefore of the utmost importance that the relationship with a testing services supplier is well balanced and healthy. Participation in each other's stakes is not allowed. Monopoly-positions (single-sourcing) should not occur. Outsourcing of testing is an activity that should not be organised on a project level. It should be set-up on a corporate level. Today many companies believe that they are performing outsourcing, but in reality they are merely using body-shopping as a means to resolve resource problems.

When a company starts to invest in professional structured testing, the company should know that they are investing in the future. Structured testing creates short-term benefits (the quality of the product under test improves) but creates a competitive advantage in the long term. This is where the real benefit is found. If companies seek for a testing partner, they should seek

for a solid partner that will help them in the long run. The external testers that are hired should not come in for that one project only. The testers should invest in their customer's business domain and become acquainted with that environment. The testers should build their testware with a long-term perspective, allowing the client to benefit from it in the future. The supplier should not fear these fundamental investments he is making, because it will give him a stable seat for the future. Becoming the "house-supplier" regarding testing assures future incomes without the need for new sales investments. If both supplier and client believe in the model, they will both benefit a lot. Setting up a strategic alliance really creates the classical Win-Win relationship. Even more, the ultimate goal gets served: the final user of the software receives a higher quality product. The result is Win-Win-Win.

The set-up of such an outsourced model requires a careful organisational architecture. Of the many elements that must be taken into account the most important and core-issue to solve is the one of responsibility. It is extremely difficult to correctly define the responsibility of the testing partner. On the one hand he must get a large authority to report defects and to advise about the quality, on the other hand, the testing partner may not become co-dependent by assuming responsibility for the end quality. The client on his side also needs to know - and more important - accept the boundaries of his authority. Furthermore, the client must guard that his responsibilities do not drift of. A typical behaviour that takes place when a testing partner is introduced is called TIM-J (That Isn't My Job). The awareness that someone will "take care of the bugs" makes the creators of the software negligent. "We don't have to check anymore, that's why we pay those expensive test experts". Consequently, the software quality goes down and the error detection and correction process takes longer and as such becomes more expensive. TIM-J is the opposite of co-dependency. The balance between these two extreme attitudes is very fragile.

Before moving to a typical example of which responsibilities should go to which party, let us first set-up a meta-model for describing the responsibilities. Many people think defining responsibilities is as simple as drawing up a list of tasks each party must do. There is, however, much more to it.

First of all, there is indeed the definition of what we call the **responsibility**. The responsibility is an area in which we are supposed to realise a certain result or performance. "It is my responsibility to clean the windows" means that I must make sure that the windows are clean. This area can be described by a list of tasks, but more preferably it should be described by a list of results.

To be able to carry a responsibility, **authority** is required. One cannot be responsible for cleaning windows if one cannot decide when and how often (and how) the windows must be cleaned. Delegating or assigning an authority is as giving the different parties a conscience, a will. They have a decision power to start or stop actions within their responsibility area.

Carrying an authority is only possible if one can measure the result of the decision taken. If you have not defined what a "clean window" means, how will you be able to decide to start cleaning or not? An authority only works if you provide both parties with an agreed **controllability**. Controllability means having a set of metrics of which both parties have agreed that they are valid and may be used to rule.

Finally, both parties must be motivated to realise the expected performance of the responsibility area. If the window cleaner gets his pay by the hours he spent cleaning, he will most probably clean a lot and very slow, regardless of the neatness of the window. It is even in his advantage to make the windows dirtier by cleaning them with wastewater, allowing him to come back over and over again. If on the other hand, he would get paid by the amount of light that can fall through the window, in other words, by the neatness of the window, he will behave

as we expected and make clean windows. Installing an incentive or motivation mechanism is called setting up **accountability**.

If any of the four elements: responsibility, authority, controllability and accountability is missing, chances are high that both parties will not have a successful relationship. The outsourcing model will fail.

Below is a typical example of an outsourcing model, taken from a live case.

CLIENT (SOFTWARE BUILDER)	SUPPLIER (TESTER)
RESPONSIBILITY	**RESPONSIBILITY**
• Write software • Repair errors found • Quality of the software	• Find errors • Quality of tests • Quality of the test advice
AUTHORITY	**AUTHORITY**
• Define the specs • Repair errors • Release software	• Report errors
CONTROLLABILITY	**CONTROLLABILITY**
• Product margin (Revenue – Support/bug fix costs) • Number of solved defects • Total project cost	• Test coverage • Test depth • Number of defects founds
ACCOUNTABILITY	**ACCOUNTABILITY**
• Product margin (Revenue – Support/bug fix costs)	• Defect Detection Percentage

Software builders are responsible for building sound products. Their products should match what they have broadcasted via their marketing departments and should also match the "de facto" norms of stability (everyone agrees that a pc that crashes every day is not acceptable).

The builders have the authority to decide what they will make, to chose the specs to which their system will comply. If they are clever, they will of course make sure that their specs matches a market need. The builders also have the authority to decide which errors, found by the testers, they will repair and which not. Since the builders are held accountable for the success of the product, they must have the freedom in deciding which defects are important and which are not.

Software builders can control the impact of their decisions by first of all looking at the success of their product in the market. The generated sales, the costs of the after sales support, all these elements can serve as metrics. The number of the defects that get solved during development is even an interesting metric. Assuming that there are errors to be found, it is wise to measure how many defects that are really solved upon releasing the software. If little solved defects are in the release, it might be correct to assume that the testing or repairing was poor. It is not very likely that there were no errors to solve at all…

Finally, software builders should be held accountable for their contribution to the bottom line. More important then the revenue generated by the product (this is more of a sales matter)

but the costs associated to support and bug-fixing activities should be used as a key to define the variable pay of the programmers and analysts.

Testers are responsible for finding bugs. This is it. Their performance is expressed in the amount of errors they find. It is of course equally important to measure their effectiveness and efficiency by comparing the testing effort (cost) to the contribution (number of defects found) they realised. As such they are also responsible for developing sound and mature testware. Test maintenance costs are partly their responsibility. Partly because the way the development is done also influences the maintenance of the testware.

If there is one clear authority that the testers have it is right to report freely the defects found. Although this sounds trivial, some companies still are very hesitant to formally report and register in a database the bugs found. Analysing consolidated defect data is sometimes not allowed. The right to report defects is essential. This right must also be enriched with escalation rules that allow the opinion of the tester to reach management level when their advice is not followed.

The real effect of the testing activities on the organisation must be measured by monitoring the real quality of the product once it has been released. For tester, the cost of support is not the metric to be used to calculate their variable pay. The Defect Detection Percentage (DDP), as developed by Dorothy Graham, serves the purpose well. DDP is the ratio of comparing the defects found by testing with the total amount of defects found (testing AND defects found in production). The fewer defects that are found in production (the higher the nominator) the more effective the testing was. DDP, at first, has little value to influence the quality of the product, but it has a high significance when it comes to analyse and thus improve the testing process. Other metrics that must steer the testing process are the test coverage: how much of the product is covered by the testing and the test depth: how much different conditions or test cases have been applied to validate a certain functionality. Much more can be said about both test coverage and test depth. This is not the intention of this paper though.

A pitfall with DDP though is that testers might argue a lot about releasing software in their eyes “too early”.

To conclude this paper, it is worth to stress that setting up an outsourced model for testing is a corporate strategical decision. One might trigger the need for outsourced testing from within a project and even implement the model on a pilot project base, but the objective must be long term and on a partner or corporate level (rather then a pure supplier relationship).

Is testing doomed to fail if one does not outsource? No, of course not, but the return of the investment in testing will be much less. A continuous effort will be required to stimulate testing in the organisation and to make sure that the testing reveals errors rather than proofs how good our products are. Personnel turnover in the test department will be huge and no learning curve effects will be experienced. Testing will never give a competitive advantage, but will become a part of the overhead cost and as such remain a necessary evil.

SPEAKER'S CURRICULUM: JENS PAS (MANAGING PARTNER)

Jens Pas is founding Partner of I2B and Managing Director - CEO. Jens Pas is an authority in the business of Software Testing. He developed a unique method for structured software testing. His method was introduced in and adopted by major Belgian and Dutch companies. He has also development Test Assessment programs and has conducted them at directors and top-management level at various sites in larger IT environments. He published various articles regarding the methodology of Software Testing and is the author of the textbook that accompanies the course "Introduction to Structured Software Testing". He is a (keynote) speaker and track chairman on many Software Engineering and Quality Conferences (Eurostar 98 Munich, 99 Barcelona & 2000 Copenhagen, **Quality Week Europe**, Fesma Brussels, Software Automation, Novi Amsterdam, Technological Institute of the Royal Society of Belgian Civil Engineers, SQE Europe Test Congress 2000). He is member of the International Advisory Board of Quality Week. Currently he is specialising in Innovation Management services.

Jens Pas was co-founder and General Manager of a Belgian consultancy company specialised in testing. Previously he was responsible for the re-engineering of the pan-European IT-support department of Quaker Oats Europe and for setting up the Application Support department of the ETS division of Barco Graphics.

Jens Pas is an engineer and holds an Executive International Master in Business Administration degree from the Vlerick Leuven Gent Management School.

Action

Reaction

Outsourcing

I2B

IDEA TO BUSINESS

Innovation Management Services

Test Outsourcing: From necessary evil to a competitive advantage

Idea to Business (I2B)

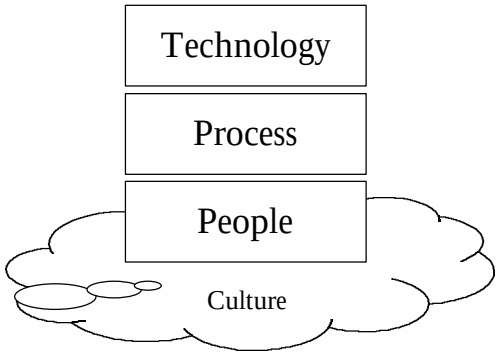
I2B cvba
Kwaadstraat 25
B-9750 Zingem
Belgium
Tel.: +32 (9) 384.86.27
Fax: +32 (9) 384.30.46
E-mail: info@i2b.be
http://www.i2b.be

What is a competitive advantage ?



Action Reaction Outsourcing I2B

The distinctive corporate capability that cannot be reproduced by the competition



Goto Start
Back
Goto End

Action

Reaction

Outsourcing

Action – We make errors

- Communication
- Software Engineering maturity
- We're only human

I2B

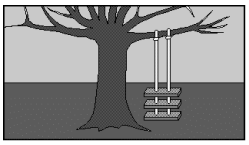
Communication

Action

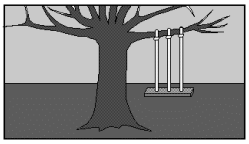
Reaction

Outsourcing

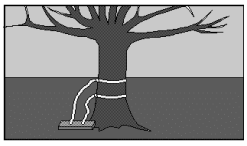
I2B



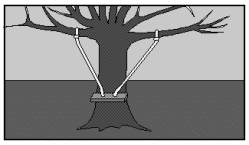
What the client wanted



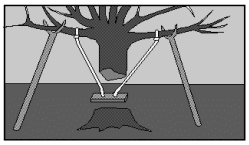
What the contract said



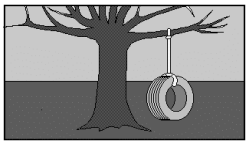
What the analyst designed



What the developer wrote



What was delivered



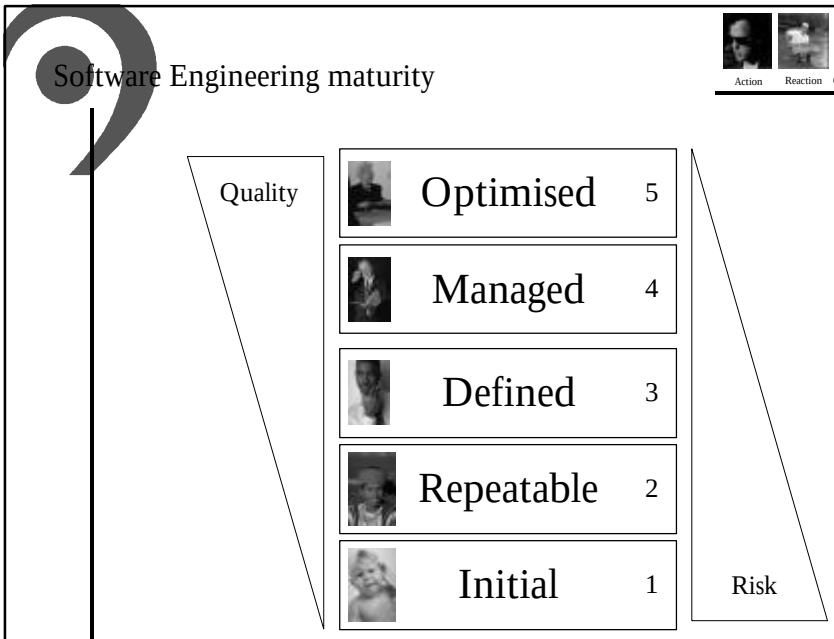
What was really needed

Goto Start

Back

Goto End

Software Engineering maturity



http://www.sei.cmu.edu/cmm/cmm.sum.html

Goto Start
Back
Goto End

We 're only human

- Memorize this:

- Limitations – Distraction
- Optimism & Subconsciousness
- Creativity

"You're having a hard time and lately you don't feel so good
You're getting a bad reputation in your neighborhood
It's alright, it's alright
Sometimes that's what it takes
You're only human, you're allowed to make your share of mistakes"

Billy Joel

Goto Start
Back
Goto End

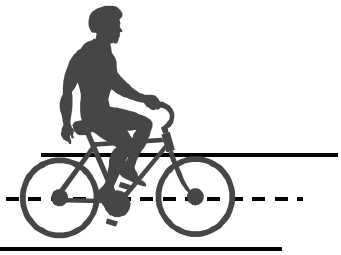
Subconsciousness

Action

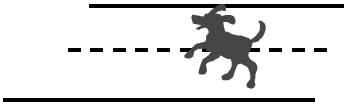
Reaction

Outsourcing

I2B



The cyclist approaches and sees the the dog



The dog in the street notices the the cyclist

Goto Start

Back

Goto End

Action

Reaction

Outsourcing

Reaction

- Preventative actions
  - Software Engineering (reduction of human intervention)
- Remedial actions
  - Testing
- The Medea tragedy
- The salmon problem

I2B

Action

Reaction

Outsourcing

Outsourcing

- Understanding Management Control
- Turning a hassle into a expertise
- Independant integration
- RACA

Understanding Management Control

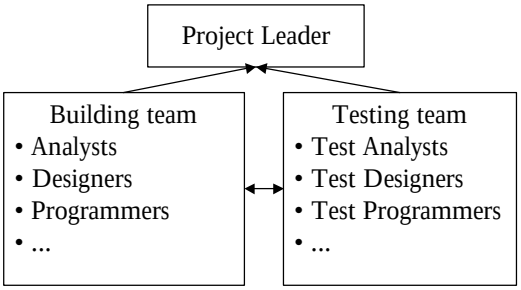


Action Reaction Outsourcing I2B

Management Control Systems
R.N. Anthony & V. Govindarajan
Irwin Mc Graw Hill, ninth edition 1998

Goto Start
Back
Goto End

Independant integration



Goto Start
Back
Goto End

RACA

	Builder (Client)	Tester (Supplier)
Responsibility	• write software • repair found defects • software quality	• find defects • test quality •quality of test advice
Authority	• define specs • repair defects • release software	• Report defects
Controllability	• Product margin • Nr. of solved defects • Total project cost	• Test coverage • Test depth • Nr. of defects found
Accountability	• Product margin	• Defect Det. Perc.

Goto Start
Back
Goto End

Thank you

Jens Pas
Managing Director – CEO
jens.pas@i2b.be
Tel.: +32 (9) 384.86.27
Fax: +32 (9) 384.30.46
GSM: +32 (475) 77.51.60

Goto Start
Back
Goto End

Our Company

I2B (Idea to Business) is a new Belgian consultancy company founded in 2000 by six people of which five are experienced consultants. Their consolidated know-how and skills have resulted in a complete portfolio of competences required to run projects concerning ICT, E-Commerce or Innovation (new business development).

Together they result in **Innovation Management Services,** offered to two types of clients: **Large Enterprises** and **Small & Medium Sized Enterprises (SME's)**. For the latter, I2B developed a special delivery model allowing SME's to receive expert knowledge to which normally only big organisations have access to.

The Mission of I2B is:

"To assure that companies can innovate and realise sustainable business from their ideas"

Goto Start
Back
Goto End

Our Credo

CREDO

We believe that our first responsibility lies with the clients who use our services. In meeting their needs our services must be of high quality and must be a reference for our clients. We cannot indulge in pressure, quantity or quick profit. We must do what we promise. We may only promise what we can do.

We are responsible towards our co-workers, the men and women who work with us. Every co-worker must be respected as an individual and must be rewarded adequatly and fairly. We must support our co-workers through a competent management, an adequate working environment and proper working conditions. Our co-workers must have the means to provide and receive feedback that allows them to learn continuously. We must support our co-workers in their family responsibilities. Our actions must be just and ethical.

We are responsible to the community in which we live. We must be good citizens, support good works and bear our fair share of taxes. We must encourage civic improvements and use our expertise to create these improvements. We must respect and protect the environment and the natural sources.

Our final responsibility is towards our stockholders. Our business must make a sound profit. We must innovate and continuously improve our methods and techniques. We must develop new services and implement them effectively and efficiently. We must create reserves to provide for adverse times. When we work according to these principles, our stockholders should realise a fair return.

Goto Start
Back
Goto End

Copyright & Liability

Copyright

The materials in this presentation are Copyright © 2000 I2B. All rights reserved . You are hereby authorized to view, copy, print and distribute these materials or parts of it subject to the following conditions:

- The materials may be used for internal informational purposes only.
- Any copy of these materials or any portion thereof must include the above copyright notice.
- I2B may revoke or modify any of the foregoing rights at any time.

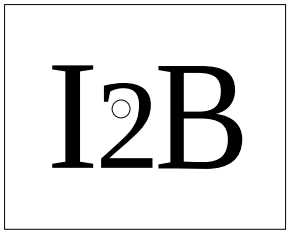
Please note that any product, process or technology described in these materials may be the subject of other intellectual property rights reserved by I2B and are not licensed hereunder.

Liabilities

The information contained in this presentation is for general guidance on matters of interest only. The application and impact of laws can vary widely based on the specific facts involved. Given the changing nature of laws, rules and regulations, and the inherent hazards of electronic communication, there may be delays, omissions or inaccuracies in information contained in this presentation. Accordingly, The information in this presentation is provided with the understanding that the authors and publishers are not herein engaged in professional advice and services. As such, it should not be used as a substitute for consultation with professional advisers. Before making any decision or taking any action, you should consult a I2B professional.

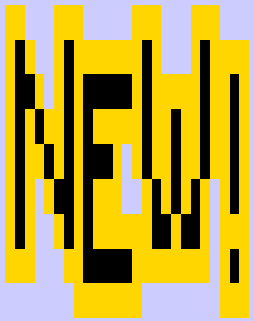
Goto Start
Back
Goto End

Idea to Business



Consultancy for those who change the world

http://www.i2b.be



QWE2000 Session 1T

Ms. Miriam Bromnick [UK]
(Ovum ltd)

"Automated Software Testing: A New Breed of Tools"

Key Points

- Automated test planning and management
- Automated requirement management
- Test case generation

Presentation Abstract

Test managers are faced with increasing problems to meet the rush to market, project timescales are getting shorter and shorter. Applications are being developed to take opportunity of new and innovative business models. More and more complex applications are being developed, using complex mixes of technology. However these opportunities bring a number of risks. Testing process needs to adapt and testers need to innovate, and to work smarter, to counter the risks.

Functional test tools are well accepted and offer good support of capture/replay testing. Scalability issues, especially for e-commerce, have meant the increasing acceptance of load and performance test tools. Even automated code checking tools have become more commonplace. However these automated test tools do not help the testing processes and they do not help test managers solve some of their biggest concerns.

New automated test tools and new techniques are required to ensure quality applications are delivered and that the rush to market does not ignore the risks of software failures.

This presentation looks at a new breed of tools that is emerging to support some of the processes around quality assurance and testing. These tools can be used standalone, or as "plug-ins" to the capture/replay tools. Some of the mainstream tools have also been enhanced to give more support in these areas, for example Mercury Interactive's TestDirector. This new breed of tools help to automate some processes that may already be used (for example as "best practice") and support application developers introducing and automating new repeatable processes. Implementing these tools will bring short term saving in delivering higher quality application and will allow long term benefits as the tools support sophisticate processes, such as requirements traceability, allowing quality across the whole life of

an application. These tools will be examined in 3 categories:

- test planning and management
- requirements management and test case generation
- inspections and reviews.

Throughout this presentation experiences of a test manager are referred to, indicating typical practical testing problems. The new breed of automated test tools is discussed in detail to understand which problems that they aim to solve. The research method used to evaluate these tools is presented, indicating the benefits of rigorous software evaluation. This presentation concludes by discussing long term investment in testing tools and ideas to influence budget holders that investment in these tools would be worthwhile.

About the Speaker

Miriam Bromnick is the editor of Ovum Evaluates: Software Testing Tools. http:\www.ovum.com\

In this role she makes in-depth evaluations of the leading software testing tool sets. She also researches into the background issues underlying testing and is currently interested in the testing of e-commerce applications. Her previous software experience spans 20 years. She followed a traditional IT career path in the commercial sector from programmer to analyst and then to project manager. She has managed testing teams in blue chip commercial organisations. As a testing and quality expert she has developed and supported software testing procedures and methods. She believes in continuous improvement and has herself recently gained a first class honours degree in Information Management.

Automated software testing: a new breed of tools

QWE2000

Miriam Bromnick

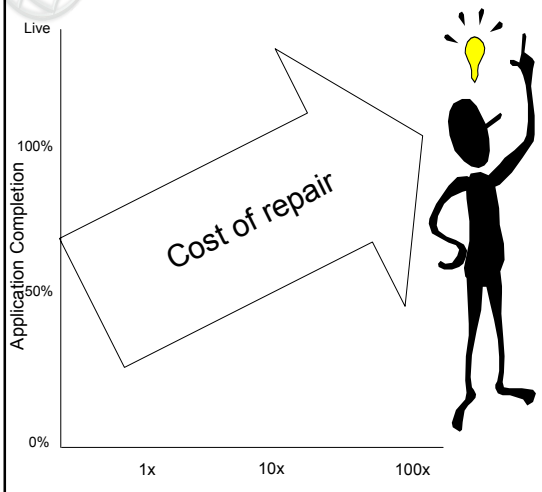
November 22nd 2000

Expert Advice on IT, Telecoms and E-commerce

Ovum

1

The cost of quality



Ovum

2

The cost of quality - 2

"Security fears shuts online bank"
*The*Guardian, front page headline

"… the bank said it followed an upgrade to its online services carried out over the weekend..."

August 1 2000

3

Ovum

Help is at hand

- **Risks and opportunities**
- Outline of some new and innovative tools
- Selecting automated testing tools

4

Ovum

About Ovum…

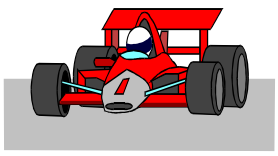
- independent research and consulting company
- expert advice on IT, e-commerce and telecoms
- "helping you to make successful decisions"
- established in 1985,
- offices in London, Boston and Melbourne
- 100 consultants provide consultancy to over 10,000 senior executives worldwide.

5

Ovum

Software development in 'internet time'

- **Opportunities**
  - ***including the e-commerce 'gold-rush'***
- Complexity and change,
  - *applications and technologies*
- **Demand for accuracy and reliability**
- **Risk of bad publicity**

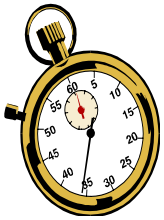


6

Ovum

Perennial problems for testing managers

- **Planning people and time needed**
- **What to test ?**
- **Runaway scope**
- **Can we stop testing yet?**



7

Ovum

More than functional testing

- Automated test planning and management
- Automated requirements management
  - *test case generation*
- Inspections and reviews
- Controlling capture/replay testing tools

8

Ovum

Help is at hand

- Risks and opportunities
- **Outline of some new and innovative tools**
- Selecting automated testing tools

9

Ovum

Automated test management and planning

- What the tools can do,
  - *T-Plan*
  - *Mercury Interactive's TD7i*
  - *Compuware's QADirector*
- What else could they suppo
  - *Test strategy definition*
  - *acceptance criteria*

10

Ovum

Automated requirements management

- What the tools can do,
  - *Caliber RM from TBI*
  - *Mercury Interactive's TD7i*
  - *Rational's RequisitPro*
  - *Compuware's Reconcile*
- What else could they support?
  - *Risk management*

11

Ovum

Test case generation

- Generating test cases from requirements
  - *Validator/req from Aonix*
  - *Caliber RBT from TBI*
- Test cases from code
  - *TestFactory from Rational Software*
  - *JTest from ParaSoft*
  - *Attol's Testware*

12

Ovum

Inspections and Reviews

- Supporting the review and inspection process
  - *ReviewPro, Caliber Tools*
- Code static analysis
  - *McCabe toolset (and others)*

13

Ovum

Controlling capture/replay tools

- What is the problem ?
- Capture/replay “test wrappers”
- What tools are out there
  - *TestFrame from CMG*
  - *Certify from WorkSoft*
  - *ActiveTest from Graphtec*
- The future ?

14

Ovum

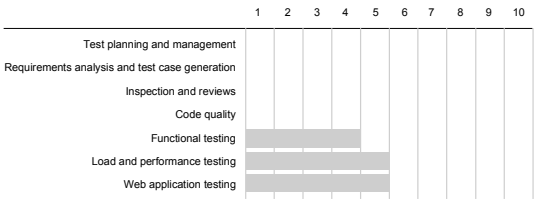
Help is at hand

- Risks and opportunities
- Outline of some new and innovative tools
- **Selecting automated testing tools**

15

Ovum

A model of automated test tools



16

Ovum

Checklists

- Use of selection criteria
- Fit for purpose
- Risks of selecting the wrong tools

17

Ovum

Risks and opportunities

- Working smarter
- Testing earlier in the life cycle
- Automating testing earlier in the life cycle
- Quote from David Gelperin

> ***"Lets stop those bugs from being born"***

18

Ovum


Miriam Bromnick
Lead Author, Ovum
Evaluates: Software Testing Tools
E-mail – mbk@ovum.com
Web – http://www.ovum.com
Tel: +44 (0) 20 7551 9169
Fax: +44 (0) 20 7551 9090
Expert Advice on IT, Telecoms and E-commerce
Ovum
19

Automated software testing: a new breed of tools

Miriam Bromnick
Ovum Ltd
Cardinal Tower
12 Farringdon Road
London
EC1M 3HS, UK

Phone: +44 (0) 20 7551 9000
Fax: +44 (0) 20 7551 9090/1
Direct line: +44 (0) 20 7551 9169

Email: mbk@ovum.com

http://www.ovum.com

The costs of not achieving quality

Everyone working in software quality and testing acknowledges the statistics that the cost of fixing a problems rises exponentially the later it is found. However there has been little automated help to achieve solving problems early on. Now a new breed of automated testing tools is emerging to help find and fix problems earlier in the life cycle.

The aim of software testing and quality assurance is to find and fix errors in the software at the point where they are cheapest to fix. Therefore it is important to:

- stop problems being built into applications
- fix problems before they get expensive to fix, for example a problem fixed during design costs a fraction to fix compared with the cost to fix during testing, let alone the expense once an application is live in the real world
- invest time earlier in the development of an application to save time over the whole life of that application.

Why now: risk and opportunities

E-commerce applications amplify the need for testing to defend enterprises against business damage resulting from software errors. The Internet has opened up a number of opportunities. New business models and new ideas to explore the full potential of the web are constantly emerging. However these new opportunities bring risks and increase the complexity of applications and the demands for rapid delivery of applications. Customers want accurate and reliable services, it doesn't matter to them what is going on below the surface.

Testing is the opportunity to make sure applications meet the needs of the customer. Testing can prevent badly performing web sites, lost business, bad publicity and even legal action resulting from software failure. Lack of testing becomes immediately visible, and so testing is changing, it is no longer:

- badly understood
- badly done
- demoted to something done at the end of development
- often delegated to end users
- denied investment in automated tools.

Along with testing becoming more sophisticated, recent research has revealed some automated testing tools that will help in the early discovery of problems. Some of these tools have been around for a few years, but now they have matured enough to be considered a new breed of tools.

Ovum's research

Ovum is an independent research and consulting company, offering expert advice on IT, e-commerce and telecoms. Our mission is to help you make successful decisions, and our analysis of key developments is highly respected worldwide for its authority, quality and clarity.

Ovum's research is provided both as consultancy and as reports. The reports are available both from the web and in paper copies. The research for *Ovum Evaluates: Software testing tools* is the basis of this paper.

Perennial problems of a testing manager

E-commerce applications are usually developed with the same constraints as other applications, however some of those constraints are amplified, including:

- demanding deadlines
- the need for extensive testing, and testing is a large-scale activity with many parts, therefore good management of the testing process is essential
- increasing complexity, therefore it is unlikely that a project will have the resources to conduct all conceivable tests
- the need for ordering the testing of the components, requiring careful planning and management.

Consequently software testing and quality assurance activities need to be well planned to get maximum assurance from the tests that are carried out.

Testing strategy

To meet the challenges of quality assurance and testing you need a testing strategy. You need to determine what, where and when you will test. The questions you need to ask include:

- What strategy will you use to prioritise the tests?
- What are the high risk areas of the application?
- What is the application supposed to do and are there any specifications?
- Where are the problems likely to be found?
- How are test results to be used?
- How many people will we need?
- How do we know when to stop testing?

All of these questions will have specific answers for specific applications. But without answers, testing will not provide an efficient way to deliver quality software.

How does the new breed of tools help

Software testing is about demonstrating that a piece of software is fit for its intended purpose. It can be split into two main categories; functional testing and non-functional testing. Functional testing tools are well accepted for repetitive tasks, such as regression testing. They work on the basis of capturing tests, scripting tests and replaying those tests. Load and performance testing tools are also frequently used to check scalability and to plan capacity.

However some more advanced automated tools are emerging to support testing and quality assurance processes, including:

- test management and planning
- requirements analysis and test case generation
- inspections and reviews
- test wrappers.

Test Management and planning tools

Current support for test management and planning

Testing needs to be built into the development cycle of any piece of software. To be productive testing need to be ordered, structured and visible. The requirements of the testing activity need to be determined, because testing without an objective wastes time and resources. The requirement has to be translated into a set of specific objectives listing exactly what you need to see from the test results. These will be refined from general, high-level objectives into test cases. Each test case needs to have a scenario that describes what will be put into it, and what the result should be. These scenarios should be re-used from the analysis and design processes. All this requires the support of planning, managing and monitoring in the same way as any other development activity.

Over the last 3 years the need for tools to support test management has been recognised by the mainstream testing tool vendors. Some test management functions are included in most of the major tool sets (for example Mercury Interactive's TestDirector and Compuware's QA Director). These support the test management process in an integrated way. However there are other automated tools in this area, that can be used alone, in conjunction with other testing tools. For example tools such as T-Plan Professional support:

- planning tests, defining criteria and cross references
- constructing of test specifications and scripts
- documenting the sequence of results
- managing the execution of tests and capturing results.

Future ideas for tools to support test planing and management

Testing of e-commerce applications increases the need to manage inter-dependent activities with tasks that may be conducted by staff in different locations and even within different enterprises. This increases the need for good planning and management. Automation is required to support identifying and monitoring the critical path through the testing activities. In circumstances with demanding deadlines it is vital that time spent on testing is

directed towards the important tasks and not squandered (for example by duplicate testing or testing interesting, but unimportant, parts of the application). Testing of e-commerce applications requires automated workflow, with all partners involved in the process sharing information about status of their activities.

In addition automated tools would be useful to support definition and documentation of test strategy. This task is often repeated from application to application and therefore the repetitive parts of this activity are ideal for automated support.

Acceptance criteria are often over-looked at the start on an application. Deadlines usually determine when testing stops rather than the achievement of specified levels of quality. Automated support for defining and documenting acceptance criteria, early in the lifecycle, would be a useful way to assist test managers in focussing the testing efforts.

Test management is still mainly conducted in an unsophisticated way and therefore it is unlikely that the vendors will see the improvement of these facilities as a high priority. We expect incremental improvements in this area. It is likely to be 3 to 4 years before the tools reach the necessary level of sophistication to provide fully automated test management support.

Requirements analysis and test case generation

Current support

Analysis and design level tools are available including case tools and visual modelling tools, such as Rational Rose, or Aonix's Software through Pictures, or DOORS from QSS. If you are using automated support of your analysis and design process then you may not need testing tools to support your requirements management process.

However the testing tools that help with requirements management will be of interest if you are struggling to keep control of the requirements. For example a basic advance would be to use spreadsheets to manually capture requirements, including those:

- agreed in meetings
- scribbled on whiteboards
- assumed by everyone on the project
- revealed informally by expert users.

The need for more formal support of requirements management has been recognised by the mainstream testing tools vendors, for example, Rational Software's RequisitePro, Mercury Interactive's TestDirector and Compuware's Reconcile.

These toolsets provide a storage mechanism for requirements and allow you to cross reference requirements to tests. More sophisticated tools are available that will:

- manage requirements
- validate requirements
- generate test cases.

These toolsets include the Caliber tools (which are based on cause-effect diagrams) and Aonix's Validator/Req (based on UML scenarios and use-case diagrams)

In addition some tools are emerging that can generate test scripts directly from the code. These include Rational Software's TestFactory, ParaSoft's JTest (for Java) and Attol's Testware.

Future support

We would like to see these tools assist in risk management and offer further help in prioritising test cases

We look forward to closer integration of test case generation with the main testing tool suites and see the need to integrate testing and quality assurance with design.

Inspections and reviews

Current support

Inspections and reviews are much neglected processes. They are techniques that can bring benefits early on in the lifecycle. Possibly one of the reasons they are neglected is due to the overheads in organising them. Therefore ReviewPro from SDT is a useful tool as it automates the organisation and reduces the bureaucracy in setting up and completing reviews. It supports a simple and effective way of implementing the review process. It can support distributed reviews across the web and TBI's requirements management tool can also support review of requirements across the web.

Several tools support static analysis of code. This gives you the ability to automate your code reviews. The McCabe toolset is the leader in this area. The static analysis tools all provide measures of the quality of the code and these metrics can be used to predict troublesome areas of applications.

Test wrappers and capture/replay tools

Current support

A few smaller vendors, from their practical experience, have perceived a need to make capture/replay tools easier to use.

WorkSoft has developed Certify, to drive the capture replay tools through business processes. CMG has developed TestFrame to drive the capture replay tools from action words. Graphtec has developed ActiveTest to help in the strategy for scripting tests from Rational Robot tests

Future support

These tools are just being developed as commercial tools for general use. We will be monitoring the market closely to see if they are indicating the future of automated test tools.

What should you do?

Develop a test strategy for process automation

To be efficient testing needs to be directed to meet the critical risks of your applications. Each enterprise needs to consider automation on a case-by-case basis. You need to develop a strategy to adopt automated quality assurance and testing tools as part of your overall management strategy. Managing the testing process provides a key to successful applications.

Automated testing

The mainstream testing tools will help you complete tasks quickly and efficiently, specifically in testing the presentation layer, regression testing and load and performance testing. Testing tools come in different shapes and sizes. You may decide to make a strategic decision to invest in tools to support your entire build processes. Or you may decide to make a tactical investment in a value for money testing tool to assist you meet your immediate deadlines.

Investing in automated testing tools provides some insurance against the risks, however just selecting a testing tool is not the solution to the problem. It is left to you to determine which risks you need to focus your testing on and whether the testing tool is taking you where you want to go in building your applications.

You need to ensure that you select tools that will fit into your culture and to plan time for training and implementation of those tools. Selection of the wrong tools can be expensive, not just the initial costs of the tool but the costs in training, implementing and supporting the tool.

Scoping your testing tool requirements

Ovum model and checklists

Ovum has a model of testing tools in 7 dimensions:

- test planning and management
- requirements analysis and test case generation
- inspections and reviews
- code quality
- functional testing
- load and performance testing
- web application testing

For each of these dimensions checklists are used to evaluate the tools. Each checklist has many criteria. From the checklists, overall scores are obtained to allow comparison between the different toolsets.

Planning for quality

You need to make sure that you plan enough time for testing and quality assurance activities. This is particularly important if you are using new tools

for the first time. Although the tools are likely to save you time in the long run you will need to plan for their implementation.

Follow best practice

Existing best practices need to be adapted to meet the demands of e-commerce and the fast rate of change in the current application development climate. Quality assurance managers and test managers now require flexibility in their approach. New tools and techniques need to be explored and piloted. Continuous improvement of testing processes is required. It is an exciting time as a new body of knowledge and test products emerges.

'Lets stop those bugs being born!'

Is the main message of this paper and is a quote from David Geleprin of SQE.

Miriam Bromnick

Biography

Miriam Bromnick is the editor of *Ovum Evaluates: Software Testing Tools*. In this role she makes in-depth evaluations of the leading software testing tool sets. She also researches into the background issues underlying testing and is currently interested in the testing of e-commerce applications. Her previous software experience spans 20 years. She followed a traditional IT career path in the commercial sector from programmer to analyst and then to project manager. She has managed testing teams in blue chip commercial organisations. As a testing and quality expert she has developed and supported software testing procedures and methods. She believes in continuous improvement and has herself recently gained a first class honours degree in Information Management.

QWE2000 Session 1A

Speaker
(Organization)

Mr. Jean Hartmann, Mr. Claudio Imoberdorf [USA]
(Siemens Corporate Research)

"Functional Testing Of Distributed Component-Based Software"

Key Points

- Structured test design based on UML state charts.
- Tool support for test generation and test execution.
- Methods suitable for testing individual components or sets of integrated components.

Presentation Abstract

Increasing numbers of software developers are using the Unified Modeling Language (UML) and associated visual modeling tools as a basis for the design and implementation of their distributed, component-based applications. Once developed, however, test cases must be designed, generated and executed to validate the components. This is especially important for unit and integration testing.

At Siemens Corporate Research, we are addressing this issue by integrating our test generation and test execution technology with commercial UML modeling tools such as Rational Rose; the goal being a *design-based* testing environment.

In order to generate test cases automatically, developers first define the dynamic behavior of their components via UML Statechart Diagrams. These views then need to be annotated with additional, test-specific information such as coverage criteria, data variations and interconnected, in the case of multiple components.

With the help of our test generation technology, test cases are then systematically derived from the annotated UML StateChart Diagrams and executed using our test execution environment, which was developed specifically for interfacing to components based on COM/DCOM and CORBA/IDL middleware.

Providing such a design-based testing environment ensures major benefits for developers:

> With minimal additional effort, developers can reuse their component designs as test specifications. In the case of code changes, these test specifications can be updated and used as a basis for automatically generating a new set of regression tests.

Developers no longer need to manually implement custom test drivers for components, which can be especially tedious and error-prone in the case of distributed components. Executable test drivers are automatically generated directly from the test specifications.

The environment can be applied to individual as well as a collection of components, making it suitable for use during unit and integration testing.

In future, we see the delivery of software components being accompanied with a standardized test specification, possibly based on UML StateChart Diagrams. This will be necessary to ensure compliance of the component as it is integrated into a larger software system.

About the Speaker

Jean Hartmann is a project manager at Siemens Corporate Research responsible for software testing technology. His research interests focus on new techniques and tools for testing components, graphical user interfaces, and internet-based systems. He received the Ph.D. degree in computer science from the University of Durham, UK, in 1993 where is thesis topic emphasized improved regression testing techniques.

Claudio Imoberdorf is a Member of the Technical Staff at Siemens Corporate Research. He has seven years experience in the area of software design, software developmemnt, and component technologies. Prior to joining Siements he was a lead designer on a component-based building automation system at Siemens Building Technology, Switzerland.

Functional Testing of Distributed, Component-Based Systems

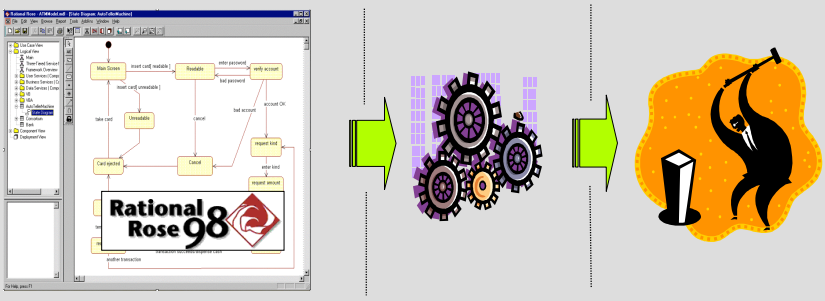
Jean Hartmann and Claudio Imoberdorf
Siemens Corporate Research
Princeton NJ 08540

Tel: ++1 609 734 3361
Fax: ++1 609 734 6565
Email: jhartmann@scr.siemens.com

Motivation

- Siemens products tend to focus on the embedded market
- Development of domain-specific frameworks within Siemens is growing, e.g. telecomms, industrial automation, energy distribution
- Most Siemens frameworks make use of COM/DCOM and are event-based rather than data-driven
- The components used in these frameworks are being developed in-house and/or purchased from third-parties
- Emphasis is on unit and integration testing of these components
- We must use standardized techniques and tools to model these components and their runtime behavior...
- …so that we can use the models to:
  - define test designs/specifications
  - derive test cases and execute them!

T*n*T : UML-based Test Generation and Execution Environment



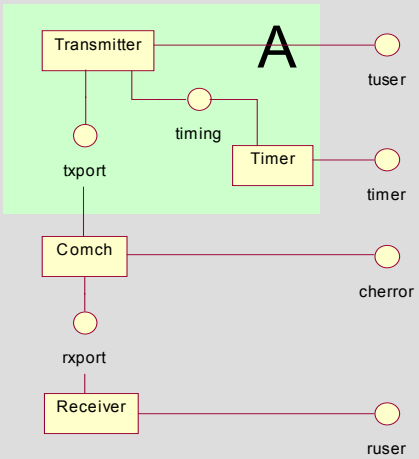
Step 1 : Modeling Component Behavior - Goals

- Component models are based on a standardized design notation => UML
- Component behavior could be modeled using either of these UML dynamic views:
  - Sequence or Interaction Diagrams (Message Sequence Charts)
  - Statecharts (our approach)
- This modeling approach:
  - focuses on black-box testing
  - supports unit and integration testing (individual/collections of components)
  - addresses COM-specific testing issues
  - aims at automating the test generation and execution steps

Step 1: Modeling Component Behavior - Definition

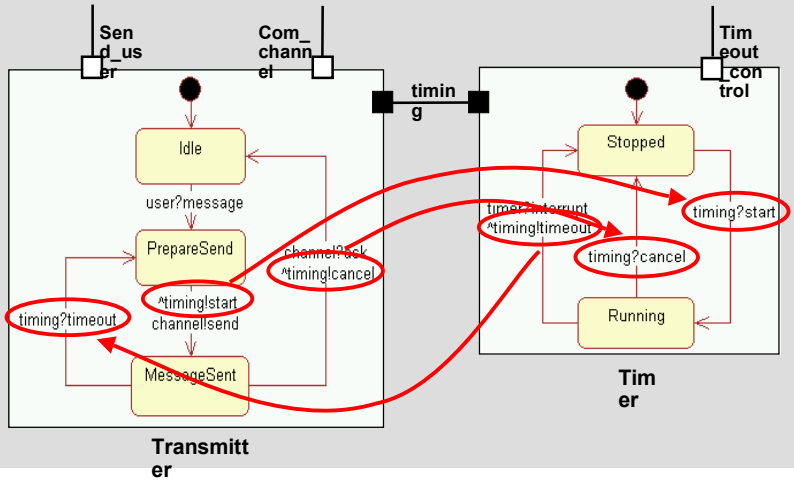
- A Statechart should express the typical and erroneous runtime behavior of a component
- For test generation, data variations of the event parameters need to be defined and specified
- To enable integration testing, a new transition labeling scheme is provided to describe the component interaction (via events)
- For integration testing, users define the collection of components to be tested (*subsystem definition)*
- Significant assumptions are being made:
  - Point-to-point communication semantics between components rather than a shared (global) event model
  - Communications are synchronous (blocking) rather than asynchronous

Example: A Communications Protocol



- Focusing on integration testing
- We want to generate test cases to validate component interaction within subsystem A - *Transmitter_Timer*
- This subsystem has:
  - External interfaces:
    - **tuser**
    - **timer**
    - **txport**
  - Internal Interfaces:
    - **timing**

Example: Describing Component Behavior and Interaction



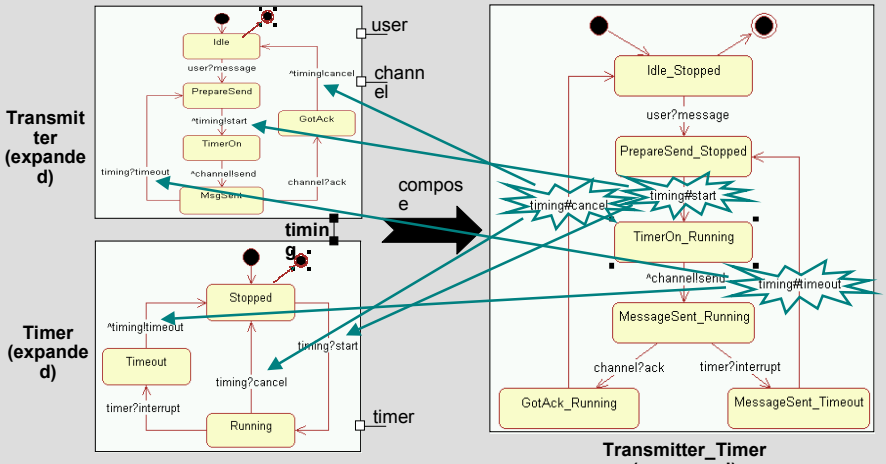
Step 2 : Generating Test Cases from Models

- The following T*n*T-internal processes are now executed:
  - extraction of the relevant data from the Rose repository (not the .mdl file!)
  - resolution of the multiple send/receive events on Statechart transitions - these are held in an intermediate model
  - computation of a global behavioral model based on matching send and receive events
    - using incremental composition and reduction algorithm of linear complexity
  - making use of any subsystem definitions defined by the user to improve scalability
- Generating test cases from the global behavioral model
  - coverage criterion: all transitions (within/between components)

SIEMENS
Siemens Corporate Research, Inc.

Software Engineering

Example: Composing a Global Behavioral Model

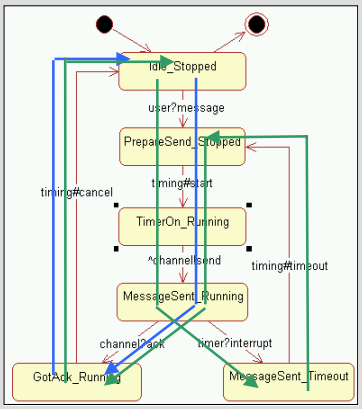


Transmitter_Timer (composed)

SIEMENS
Siemens Corporate Research, Inc.

Software Engineering

Example: Test Case Generation for Subsystem A



Transmitter_Timer (composed)

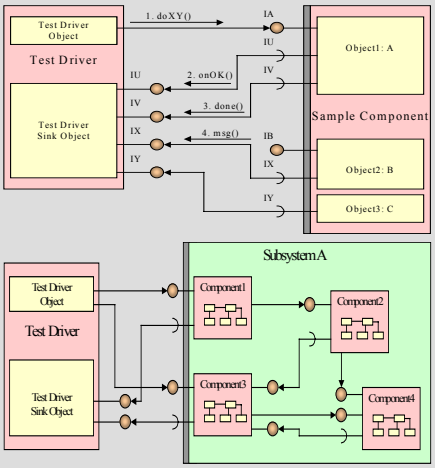
```
TEST CASE #1
 * IN  user.message();
 * OUT channel.send
 * IN  channel.ack();
 * IN  user.message();
 * OUT channel.send
 * IN  channel.ack();
TEST CASE #2
 * IN  user.message();
 * OUT channel.send
 * IN  timer.interrupt();
 * OUT channel.send
 * IN  channel.ack();
 * IN  user.message();
 * OUT channel.send
 * IN  channel.ack();
```

Test Cases for Transmitter_Timer

SIEMENS
Siemens Corporate Research, Inc.

Software Engineering

Step 3: Executing the Generated Test Cases

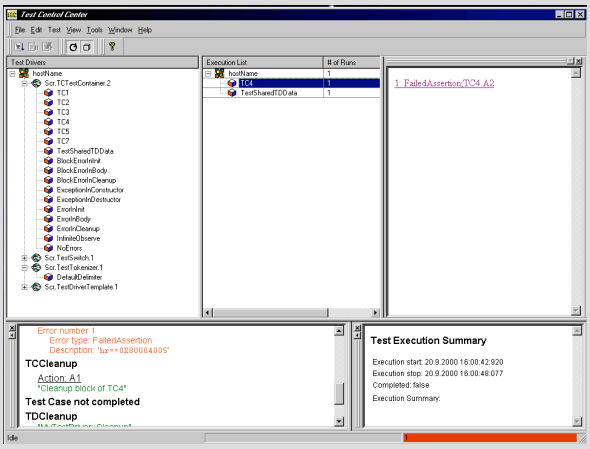


- Test execution requires the component under test to be stimulated and its response verified
- Verification in this context means : which events are being received and when?
- These events can be either synchronous or asynchronous
- We developed an event pattern matching language (ITL) to help with verification

SIEMENS
Siemens Corporate Research, Inc.

Software Engineering

Step 3 : TECS - Test Exeution Tool for COMponents



- TECS provides:
  - test harness library
  - sink generator
  - test execution monitor (TCC)
- It can be run as a standalone tool or as part of T*n*T
- It is integrated into Visual C++ 5.0/6.0
- Supports automated regression testing during unit and integration testing

Summary

- We are developing a UML-based test generation and execution environment for COMponents
- It is a unique toolset - no other commercial testing tools provide these features
- Targeted at event-based systems developed in C++/COM under Windows NT
- We have started to apply it within Siemens, e.g. SIPLACE Pro product framework - the component modeling is being completed
- We are continuing to refine and improve T*n*T…

- *A paper is available - it was presented at the International Symposium on Software Testing and Analysis (ISSTA 2000)*

QWE2000 Session 1I

Dr. Lingzi Jin [UK]
(FamilyGenetix Ltd.)

"Introducing Quality Assurance Into Website Development: A Case Study For Website Quality Control In A Small Company Environment"

Key Points

- Website testing process
- Test planning and design
- Automated web testing

Presentation Abstract

The rapid development of internet technology and e-commerce gives opportunities to many small companies who are developing various websites. Most of them do not have quality control in place. This paper is divided mainly into two parts. Firstly, it describes real world experience of setting up quality assurance and introducing best practice of website testing into such companies. The second part shows how formal testing techniques are applied in test case design and automated testing tools are used for the development of a website GeneAlert for genetic risk analysis. Familygenetix was previously a software publishing company specializing in software for genetics, chemistry and mathematical modeling. It now converted business focus to provide service for health care and software development for genetics research. All software development is in house. It started with somewhere between "initial" and "phase definition" levels of software testing [1]. Though software testing is defined as a phase that follows coding and testing is separated from debugging, test planning and test data preparation is done without proper requirements specification and post-code, execution-based testing is considered the primary testing activity.

About the Speaker

Dr. Lingzi Jin graduated from Department of Computer Science, Nanjing University, China, in 1982. She had her PhD degree in Software Engineering from Department of Computer Science, Nanjing University, China, in 1987. Lingzi Jin is a software test manager of FamilyGentix Ltd. Before she joined FamilyGenetix in Feb. 1999, she worked in academic for about 10 years teaching and doing research in software engineering. She is the co-author of about 30 papers and books. She is currently interested in applying formal testing techniques to industrial scale software, software process improvement and measurement of software quality.

Quality Assurance and Testing Web-based Applications
— A case study in a small company environment

Lingzi Jin

FamilyGenetix Ltd,
The Magdalen Centre, Oxford Science Park,
Oxford OX4 4GA, UK
Email: lingzi@familyGenetix.com

Outline

- Introduction
- Setting up quality assurance system
- Testing FamilyGenetix risk assessment service
- Conclusion

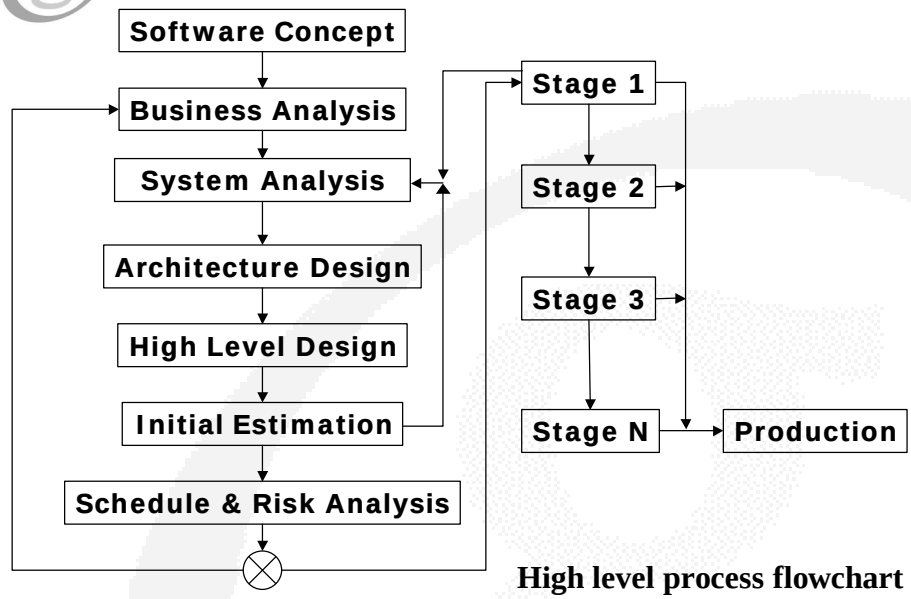
Problems of software development in small companies

- Lack of rigorous and stable development process
- Lack of documentation
- Lack of control on change requests

Setting up a quality assurance system

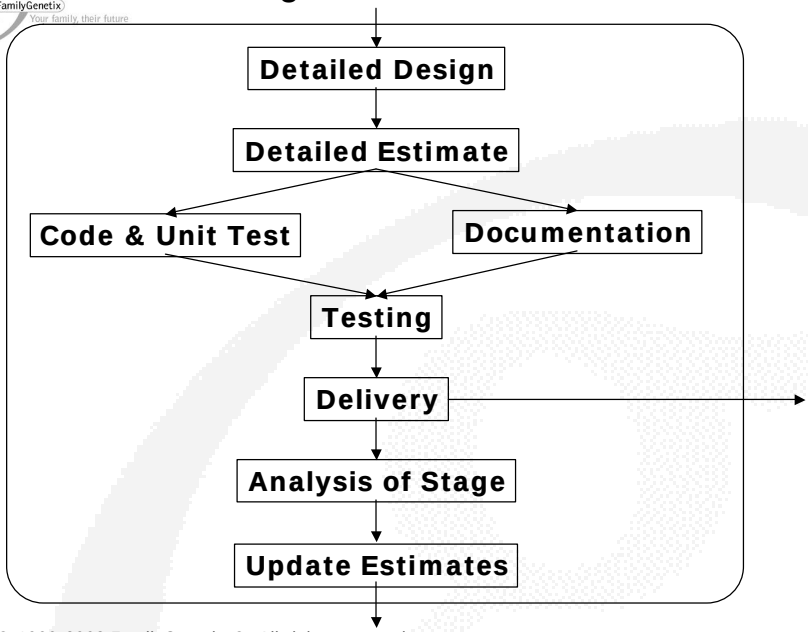
- Background of the company
- The quality improvement process
  - Introduction of a configuration management system
  - Introduction of a pre-defined development process
  - Introduction of automated software testing tools
  - Definition of testing process

The development process



High level process flowchart

Stage flowchart



12-step testing process (1)

Step 1. Requirements phase testing by document review or walkthrough

Step 2. Design phase testing by document review or walkthrough

Step 3. Perform risk analysis

Step 4. Establish test objectives

Step 5. Construct test plan

Step 6. Programming phase testing by unit and integration testing and code walkthrough

12-step testing process (2)

Step 7. Build system test data and execute tests manually or automatically and report results

Step 8. Regression testing and defect tracking

Step 9. Acceptance testing

Step 10. Installation testing

Step 11. Software change (maintenance) testing

Step 12. Measurement and evaluation of QA effectiveness

Testing Web-Based Applications:

FamilyGenetix risk assessment service

- Background: the application
- The testing process
  - Test Plan
  - Test case design
    - GUI testing using formal state transition rules
    - Functional testing using task analysis to generate scenarios
  - Usability test

GUI testing

State transition rules:

Specifies interface ***operations,*** *e.g.*

{breast cancer = checked,
age-for-breast-cancer = 40-44,
press (Next)}

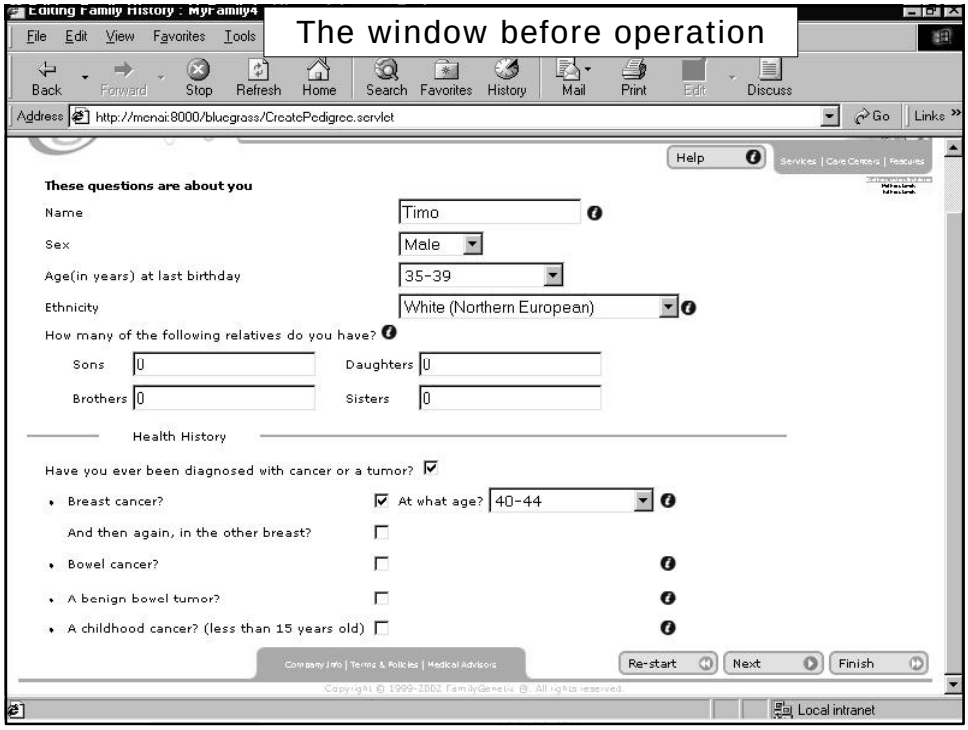
[pre-condition] {events} [post-condition]

Specifies the window ***before*** *operations, e.g.*

[name =Timo, sex =male, age =35-39, ethnicity = White (Northern European), sons =0, daughters =0, brothers =0, sisters =0, cancers =checked, breast cancer = unchecked, bowel cancer = unchecked, benign cancer = unchecked, childhood cancer = unchecked]

Specifies the window ***after*** *operations, e.g.*

[new msgbox (“Manifestation age is greater than current age”]



FamilyGenetix
Your family, their future

Functional testing

Step 1. The selection or generation of a set of task scenarios

Step 2. The analysis of the system's behavior on each task scenario to get an activity list

Step 3. Executing the test

Scenarios and task analysis

Tasks for FamilyGenetix risk assessment services:

- Risk assessment for breast cancer
- Risk assessment for ovarian cancer
- Risk assessment for bowel cancer
- Risk assessment for hypercholesterolemia
- Edit family pedigree information

Scenario generation -- an example

"The individual has an increased risk of developing breast cancer if he/she has 3 or more affected first or second degree relatives on the same side of the family, regardless of age at diagnosis."

- 3 affected first degree relatives on the paternal side
- 3 affected first degree relatives on the maternal side
- 3 affected second degree relatives on the paternal side
- 3 affected second degree relatives on the maternal side
- 1 affected first degree relative and 2 affected second degree relatives on maternal side
- 1 affected first degree relative and 2 affected second degree relatives on paternal side ...

Test case and activity list generation

Scenario:
e.g. “3 affected first degree relatives on the maternal side”

↓

Detail scenario:
e.g. (mother, daughter, sister), (mother, brother, son), ...

↓

Test cases:
e.g. Family pedigree diagrams

↓

Activity list:
Step-by-step interface operation events

Usability testing

Usability testing

- The protocol
  - video started recording
  - the user sat at computer and put on microphone
  - the user was asked about how familiar with computer, Internet and healthcare
  - some exercises were given to enter a specific family pedigree to assess risk
  - the user was asked to answer some questions of the exercise
- The main result
  - nearly every user used the system successfully
  - users said it was easy to use the system

Conclusion

- Set up quality assurance system in a small company
- 12 step testing process
- Formal functional/GUI testing of web application
- Usability testing

Quality Assurance and Testing Web-based Applications

— A case study in a small company environment

Lingzi Jin

FamilyGenetix Ltd, The Magdalen Centre, Oxford Science Park,

Oxford OX4 4GA, UK, Email: lingzi@familygenetix.com,

Abstract

Quality assurance in software development has been studied in the context of large IT enterprises, but less well understood in the context of small software companies. Existing theories and techniques such as the Capability Maturity Model have been proposed for large-scale companies. Such techniques may not be suitable for the majority of small software companies. However, their survival in the intensive competition in the IT industry heavily depends on their ability to deliver quality products and services on time. This paper describes a real world experience in setting up a cost effective quality assurance system and introducing best practice of website testing into a small-scale software company. It also shows how formal testing techniques are applied in test case design and how automated testing tools are used for the development of a website for genetic risk analysis.

Keywords: quality assurance, website testing process, test planning and design

1. Introduction

The rapid development of Internet technology and e-commerce gives opportunities to many small companies who are developing various types of websites. However, as competition in the brutal IT industry gets more and more intensive, quality assurance is becoming a crucial issue for these companies to survive. Unfortunately, most of them do not have any quality assurance system in place. It is observed that small software companies often suffer from the following problems in the development of software.

- Lack of rigorous and stable development process

There is no pre-defined software development process for the development teams to follow. Development activities happen randomly without well-established plan. A common practice in such an environment is to start the development process with some sort of interface design, then jumping into coding, without requirements analysis and specification, without any software architectural design and evaluation, nor usability analysis. Once an architectural or design problem is found in later stage of development, it is very difficult to change the system. The success of a project solely depends on the key member(s) of the project.

- Lack of documentation

The intensive pressure on a development team to deliver a software system within a very limited period of time often results in cutting off documentation. It is often the case that the only thing a

tester gets is a piece of software code without any documentation about what it is supposed to do. A tester may have to rely on intuition and software development and testing experience and to get some explanation from the developers when he doubts the correctness of the software. However, explanations are often vague and incomplete. Because functionality is not precisely defined in documents, there is no clear distinction between "feature" and "bug". Whether the software crashes is often used as the hard evidence of bugs and the judgment for fixing them. Moreover, the tester may be given no time for test planning because he has no chance to understand the software before he really sees the software running. Once he gets an executable software system, the delivery deadline is already approaching, which leaves little time for careful planning and test case design.

- Lack of control on change requests

Changes of code, and even design, may be dealt with in an ad hoc way without proper control and careful consideration of what should be a better solution. Developers may pick up whatever bugs they think they can fix and make changes to the code. Such changes neither take risk profiles into account, nor properly notify other people involved in the development. In particular, technical writers and testers can be unaware of what has been changed.

Potential consequences of such problems are the late delivery, increased cost and unstable quality. Late delivery results in loss of market share and less productivity. Cost will be increased if the development is behind schedule. A product full of bugs may seriously damage the company's reputation and the market share of all of its products. With the increasing competition in the IT market, quality assurance is becoming a crucial issue related to the survival of small software companies. Therefore, a quality assurance improvement process is essential for the survival and growth of small software companies. Unfortunately, there is no such well-established theory and technique ready to use for those companies. Existing methods in the literature such as the Capability Maturity Model are more suitable for large-scale enterprises.

This paper describes a real world experience in setting up quality assurance system and introducing best practice of website testing into such a company. It also shows how formal testing techniques are applied in test case design and how automated testing tools are used for the development of a website for genetic risk analysis.

2. Setting up quality assurance system

2.1. The Company

The company in our case study is FamilyGenetix. It was set up in 1990 as Cherwell Scientific, a software publishing company specializing in developing and marketing scientific software for genetics, chemistry and mathematical modeling. Within 10 years, it successfully developed and marketed several products for pharmaceutical and health care markets. It has now changed business focus to provide services for health care through the Internet while continuing to develop software for genetics research. All software development is in house. As with most small software companies, the number of technical staff fluctuates. The process of setting up a quality assurance systems with the company started about one and half years ago. At that time, its level of quality management was somewhere between "initial" and "phase definition" [1]. Though software testing was defined as a phase that follows coding and separated from debugging, test planning and

test data preparation was done without proper requirements specification and design documentation. Post-code execution-based testing was considered as the primary testing activity.

Having realized the importance of quality assurance and learnt from past experiences and lessons, the move to set up development process and quality assurance process started.

2.2. The process of improvement

The first step of introduction of a quality assurance system is the introduction of a configuration management system. In the selection of a tool from a large number of systems available on the market, a long-term quality improvement plan was in mind. In particular, the following requirements were considered. Firstly, it must be easy to use by the software developers to reduce the resistance to introducing quality management mechanism. It must let software developers see the benefit of using such a tool. Secondly, it should tightly integrate version control and change request so that the tool can deal with quality problems due to uncontrolled modifications. Thirdly, it should support defect tracking because it is a key issue in software quality management. After evaluation of several similar candidates, StarTeam was chosen because of its easiness of use and tight integration between version control and change request. The introduction of the tool was an immediate success. It was welcomed by the developers, tester and managers. It significantly reduced the confusion caused by using different versions, reduced the work on reporting errors found in testing and tracking the work on fixing bugs. However, the use of configuration tool does not solve the problem of documentation and modification control.

The second step was the introduction of a pre-defined development process. A modified staged-delivery development process was adapted from RAD development process [2]. It was approved by the company's management and agreed by development teams. High-level process flowchart and stage flowchart are illustrated in Figure 1 and Figure 2, respectively. This process was believed suitable for the company because of its following advantages.

- Progress is easier to track.
- Estimates can be successively refined.
- Critical functionality is available earlier.
- Problems become evident and can be assessed earlier.
- Risks are reduced early in the development cycle.
- Provides a balance between flexibility and efficiency.

The modifications of the process model were also made to suit to scale of the company. They are (a) the addition of an estimation step after high level design and an estimation step in each stage, and (b) addition of a major decision point after risk review. They are intended to address the weakness of small companies, as they are financially weak and unable to sustain high risks and loss.

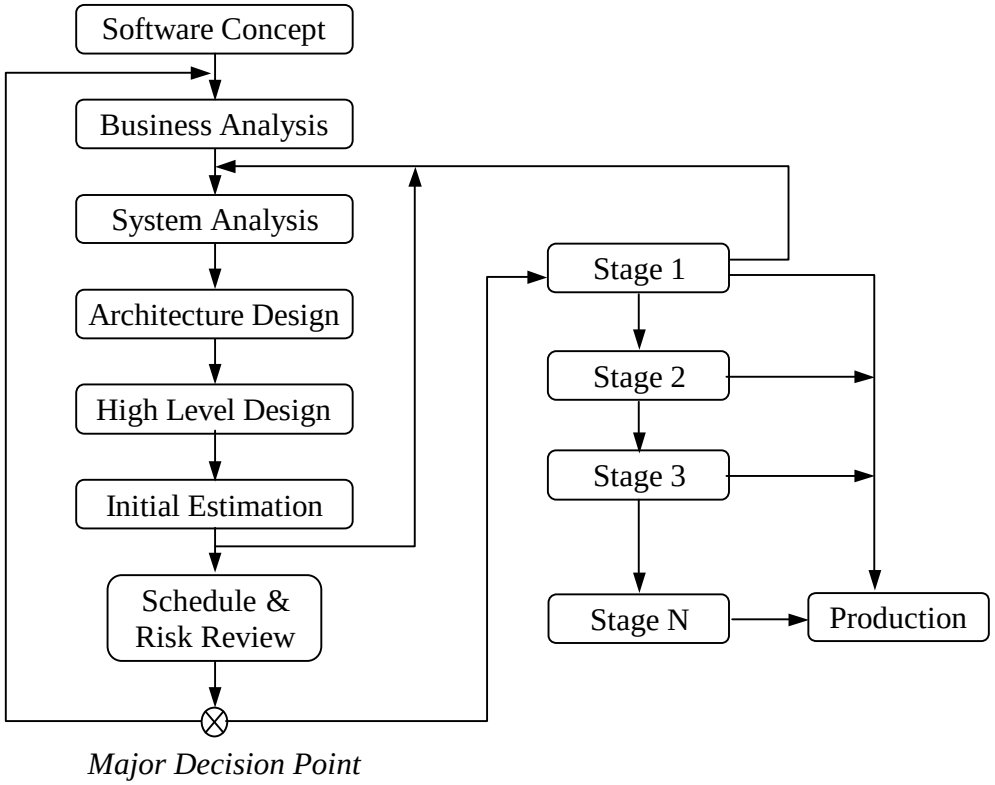


Figure 1. High-level process flowchart

Process flow matrix and documentation standards were also defined as part of the process model.

The implementation of the process model was much more difficult than the first step. Its success was less obvious and self-evidence. When it was started, some developers still considered documentation as a burden and secondary to coding. Sometimes documents still came after coding. However, the production of requirements specifications at earlier stage enabled the testing team to plan the testing work properly and to perform testing more efficiently and effectively. As a consequence, products tend to become stable in less time with less major revisions.

In parallel with the definition of development process model, automated software testing tools were introduced and used in the development. In the selection of a tool from a number of similar candidates available on the market, the long-term view of quality improvement was in mind again. The following requirements were particularly considered. Firstly, the tool must be able to support both the current testing tasks, such as automated GUI testing. Secondly, it should also support testing tasks in the near future requirements, such as web load testing, which was not required at the time of evaluation of the tool but soon became a daily testing activity afterwards. Finally, it should have the potential of supporting the relatively long-term future testing requirements, such as supporting testing in the whole software development life cycle like testing at requirements and design phases. In view of the full implementation of development process model, testing at requirements and design phases will become an important part of quality assurance activities. The use of automated testing tool successfully reduced workload on GUI testing and improved the testing efficiency. However, the strength of the tool was unable to be fully realized due to the

frequent change of interface design. Every time changes to the interface were made, many scripts have to be re-written/re-record, though some techniques can be used to mitigate this problem. This has become a large part of testing task.

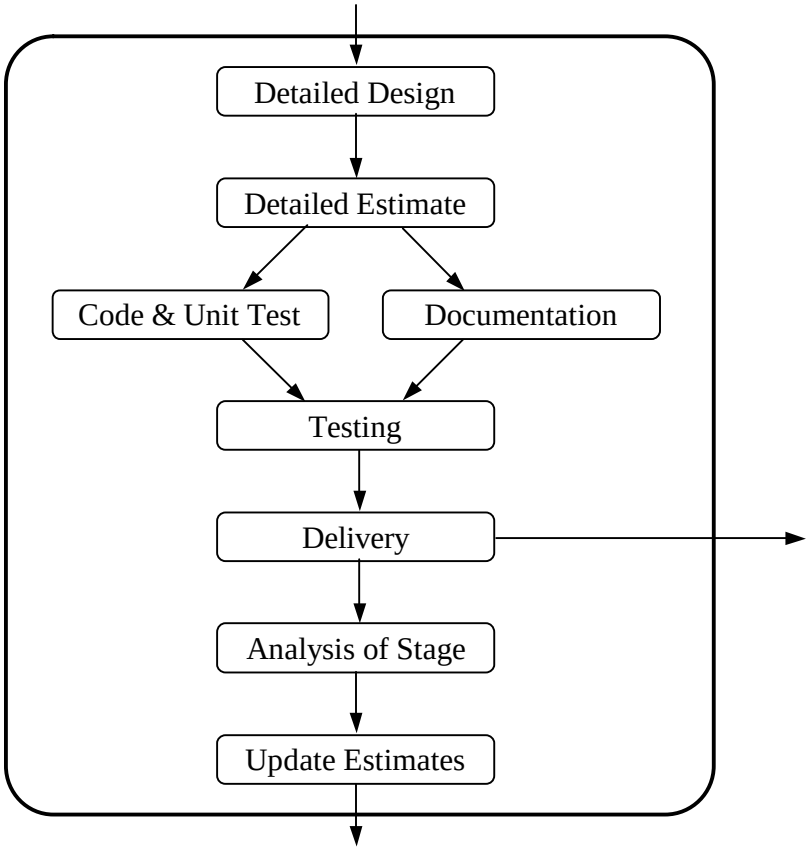


Figure 2. Stage flowchart

The third step in the process of setting up a quality assurance system was the definition of a detailed testing process. Based on existing software testing processes proposed in the literature such as [3,4], a testing process was also defined, which consists of 12 steps to be explained in the next section. The implementation of this testing process has not completed, yet. It will be synchronized with the implementation of the development process model.

2.3. Defining testing process

Testing in small-scale software companies must be suitable to the following characteristics of such companies. In particular, a small company has very limited resource available for spending in testing. Therefore, testing cannot be done thoroughly on as many aspects and/or components as it could be in a large enterprise. It requires more careful planning and trade-off between quality and time-to-market. Secondly, the development process may not be as rigorous as it might be in a matured large enterprise. As a result, the documents and information that a formal testing method depends on may not be always available. Hence, the testing process must be flexible to be able to handle such situations. On the other hand, small companies also have advantages. For example, testers often work closely with the developers in a friendly environment. Communications between

the testers and developers can happen anytime in many different forms informally or formally. Such communications are of vital importance for mutual understanding of the roles in quality management and the ease of exchange of knowledge about the application domain, the system's designs and their rationales that are difficult to document in written formats. Based on these features, we defined a testing process that consists of 12 steps. Each step is explained as follows.

(1) Requirements phase testing by document review or walkthrough

Experience has shown that the requirements phase is the most cost-effective phase in which to detect a system flaw. It should be made sure that the defect at this point will not be incorporated into the design and coded into a program. The primary objectives of testing requirements are to:

- Determine that the requirements fairly represent what the user needs.
- Determine that the needs have been defined and documented.
- Determine that the business problems have been solved
- Verify that the control requirements have been specified.
- Verify that a reasonable alternative was selected among the most probable alternative solutions.

(2) Design phase testing by document review or walkthrough

The user and IT personnel normally carry out this type of walkthrough. The design deliverables are inspected. The review team looks for three types of defects: (a) errors, meaning something has not been put correctly, (b) missing, meaning something that should have been put in, but was not; and (c) extra, meaning something not intended was changed or added.

For both requirements and design phase testing, guidelines and checklists should be developed to ensure all the deliverables reach high standard.

(3) Perform risk analysis

The main tasks of risk analysis are: (a) to identify the most important business functions as high-risk components that must be tested thoroughly; and (b) to identify certain error-prone components specific to the application that also must be tested rigorously. This analysis can be based on documentation about requirements, business model/functions and business risk assess and past development experience. Risk analysis will enable a test manager to allocate test resources in priority of risk profiles, i.e., the high risk parts get the most test effort, medium-risk parts get less, and minimal resources for low-risk testing areas.

(4) Establish test objectives

A test objective is a statement of what the tester wants to accomplish when implementing a specific testing activity. Each testing activity may have several objectives and there are two levels of objective specification. A test plan should contain high-level general objectives in the overview section and specific low-level “provable” objectives for each particular type of testing being

implemented. Test objective priorities and completion criteria should also be given in this step. Staffing and resource allocation can be decided at this stage, though, as a small company, only one or two testers are available. Split of tasks is actually not so hard.

(5) Construct test plan

The purpose of the test plan is to specify the test design, test construction, test execution, and test analysis process. The test plan also describes the test environment and required test resources. It should provide measurable goals by which management can gauge testing success and facilitate communications within the test team, the test team and the development team, and between the test team and management.

(6) Programming phase testing by unit and integration testing and code walkthrough

Sufficient self-testing work should be done by the developers and among the development team before the software is passed to testers. The code walkthrough should cover things such as: program complies with methodology, program conforms to design, program is maintainable, and more detailed things such as data integrity controls implemented, authorization rules implemented, security procedures implemented, and operating procedures developed.

(7) Build system test data and execute tests manually or automatically and report results

Testers prepare data in files or generate some database using an automated testing tool, decide what the output would be, run the software under test, analyze processing results and enter bug reports into a bug database. In the case of automated testing, scripts should be planned and recorded. They can then be maintained and run repeatedly.

(8) Regression testing and defect tracking

When change requests are raised and fixing has been done, regression testing will be run to verify that the software works correctly. At the end of regression testing, make sure all major and important bugs have been fixed before it is signed off for release to the user.

(9) Acceptance testing

Theoretically speaking, this is the final check by the users to confirm that the software functions as they require. Many users may not have the skill sets needed to perform a proper acceptance testing. They need IT professional's help to develop acceptance criteria, test cases and reports, and finally reach an acceptance decision.

(10) Installation testing

Installation places a system under development into an operational status. Concerns needed for testing are things such as: installation complies with methodology, integrity of production files is verified, accuracy and completeness of installation are verified, and documentation is complete.

(11) Software change (maintenance) testing

This type of testing occurs after the software has been placed into production. The work involves update of the test plan/data, running regression testing, and in particular, checking documents changed accordingly. Automated testing scripts are particularly useful to make sure the changes did not break the other parts of the software, if they are maintained properly.

Change control should be carried out before any change occurs. A change control board meeting should be held to decide what kind of changes should be made and to estimate risks.

(12) Measurement and evaluation of effectiveness of QA

This activity consists of two parts: first, it evaluates the performance of individuals conducting the test; and second, the results of the evaluation can be used to modify the test process. The objective of assessment is to identify problems so that corrective action can be taken.

3. Testing web-based applications

In this section, we take an example of testing a web-based application to illustrate the testing planning and techniques.

3.1. The application

The real example discussed in this section is FamilyGenetix risk assessment service. It provides Internet-based online services to healthcare professionals and the public through insurance and healthcare organizations, enabling them to assess their risk of being affected by different inherited medical conditions. We currently support risk assessment for breast cancer, ovarian cancer, bowel cancer and family hypercholesterolemia. The development of the system has followed the process model presented above.

3.2. Test planning

During the planning phase in testing the system, the following issues are considered:

- Compatibility testing

Since the system may be accessed through the Internet, the users may use the system via various different WWW browsers. These browsers are not as compatible as they should be. Therefore, to ensure the system can be used with by most users, the compatibility for different versions of browsers are tested.

- GUI testing

GUI testing checks that window objects and characteristics, such as menus, size, position, state, and focus, conform to standards. Navigation checks also belong to GUI test.

- Functional testing

Functional testing verifies the application by interacting via the GUI and analyzing the result. The goal of this testing is to focus on the requirements that can be traced directly to the functional

requirements and verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the functions.

- Performance and load testing

Performance testing and load testing verifies the performance behaviors of the system under both normal anticipated workload and anticipated worse case workload. In particular, it verifies that the system can sustained the maximum number of clients connected to the system when a large database size has been reached to perform the same functions for an expected period. This is an important issue for a system that provides the service to the public via Internet.

- Usability testing

Usability testing evaluates the ease of use by experienced and inexperienced computer users.

- Data and database integrity testing

Data integrity testing ensures that database access methods and data processes function properly without data corruption.

- Security and access control

Personal health information is private information, hence highly confidential. However, Internet-based applications are vulnerable to hackers' attacks. So security and access control are very important.

- Failure/Recovery testing

Failure and recovery testing evaluates system's ability to deal with failures. It verifies that the designed recovery processes properly restore the database and system to a desired, known state.

3.3. Test case design

Having identified test requirements, test cases are designed to meet each test requirement. This part describes the techniques to develop test cases for testing GUI intensive applications.

3.3.1. GUI testing using formal state transition rules

In a website, functionalities are triggered by operations like mouse clicking and keyboard entering. A state transition representation of the system was derived from its design. It uses the pre/post-condition-like formula in the following form to formally describe state changes before and after these operations.

[pre-condition] { events } [post-condition]

For example, a user can enter information in a window shown in Figure 3. The state of the system as shown in the window can be described as follows, where each item in the form of X = Y means the attribute X is of value Y and each attribute indicates a control in the window with which user can input information.

[name =blank, sex =female, age =select age range, ethnicity = please select, sons =0, daughters =0, brothers =0, sisters =0, cancers =unchecked]

The user may enter information in controls to change the value of the attributes. For example, the following specifies events where the user enters 'Timo' to the control indicated by 'name', enters 'male' to the control 'sex', '35-39' to 'age', and so on.

{name = Timo, sex= male, age = 35-39, ethnicity = White (Northern European), sons=3, brothers = 2, sisters =1, cancers =checked}

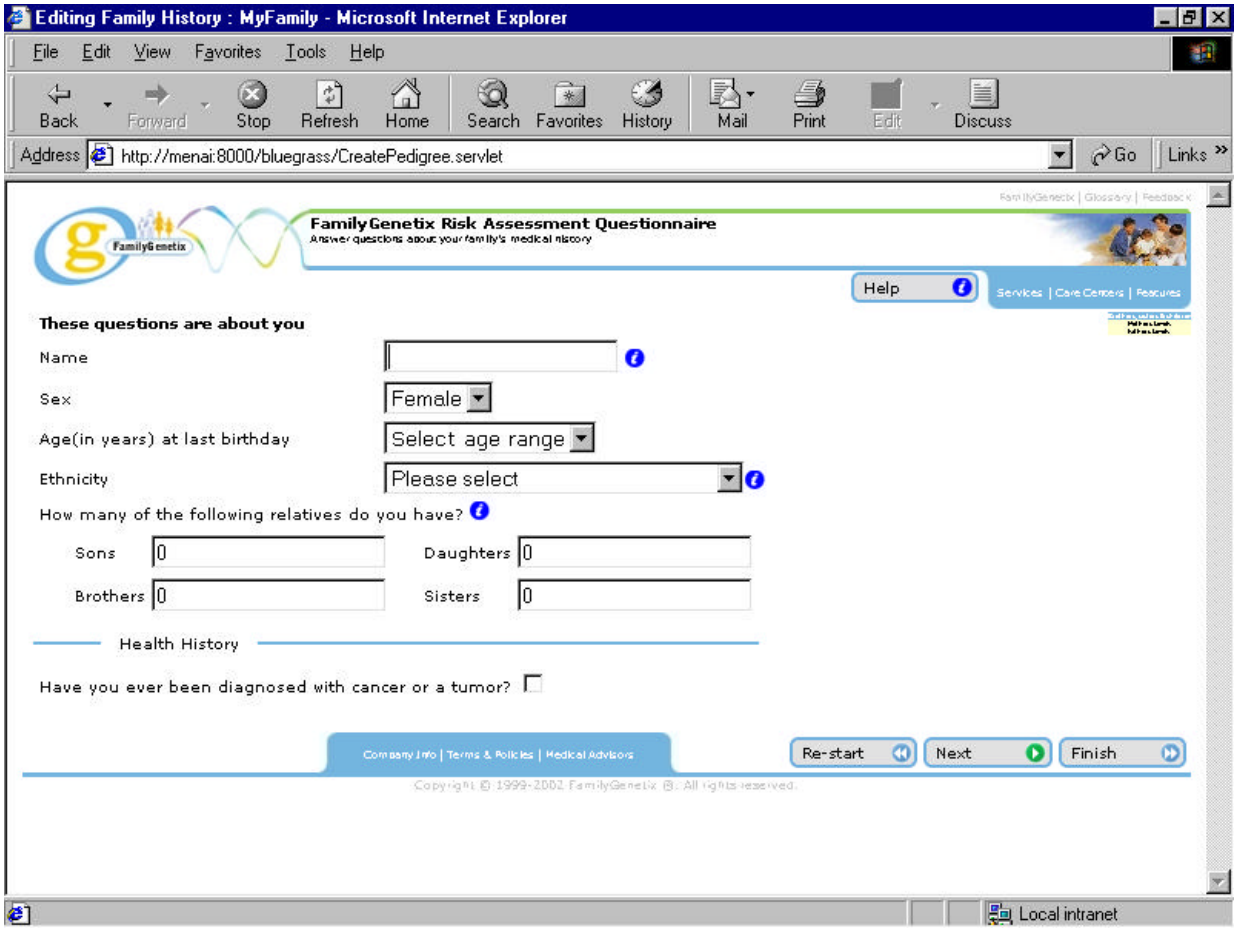


Figure 3. Window before the event of entering information

After these actions, the states of controls in the window can be described as:

[name = Timo, sex= male, age = 35-39, ethnicity = White (Northern European), sons=3, daughters=0, brothers = 2, sisters =1, cancers =checked, breast cancer = unchecked, bowel cancer = unchecked, benign cancer = unchecked, childhood cancer = unchecked]

Assuming that the information entered by the user does not change after the event, those attribute - value associations defined by user's action need not to be repeated in the post -condition. Therefore,

we simplify the representation of the post condition by only including those attribute -value associations that are not just defined by the user. For example, the above post -condition can be equivalently expressed as follows.

> [breast cancer = unchecked, bowel cancer = unchecked, benign cancer = unchecked, childhood cancer = unchecked]

A system's behavior can be formally specified by a set of such state transition rules. They are formal and easy to understand. Moreover, each state transition rule in this form directly corresponds to one test case.

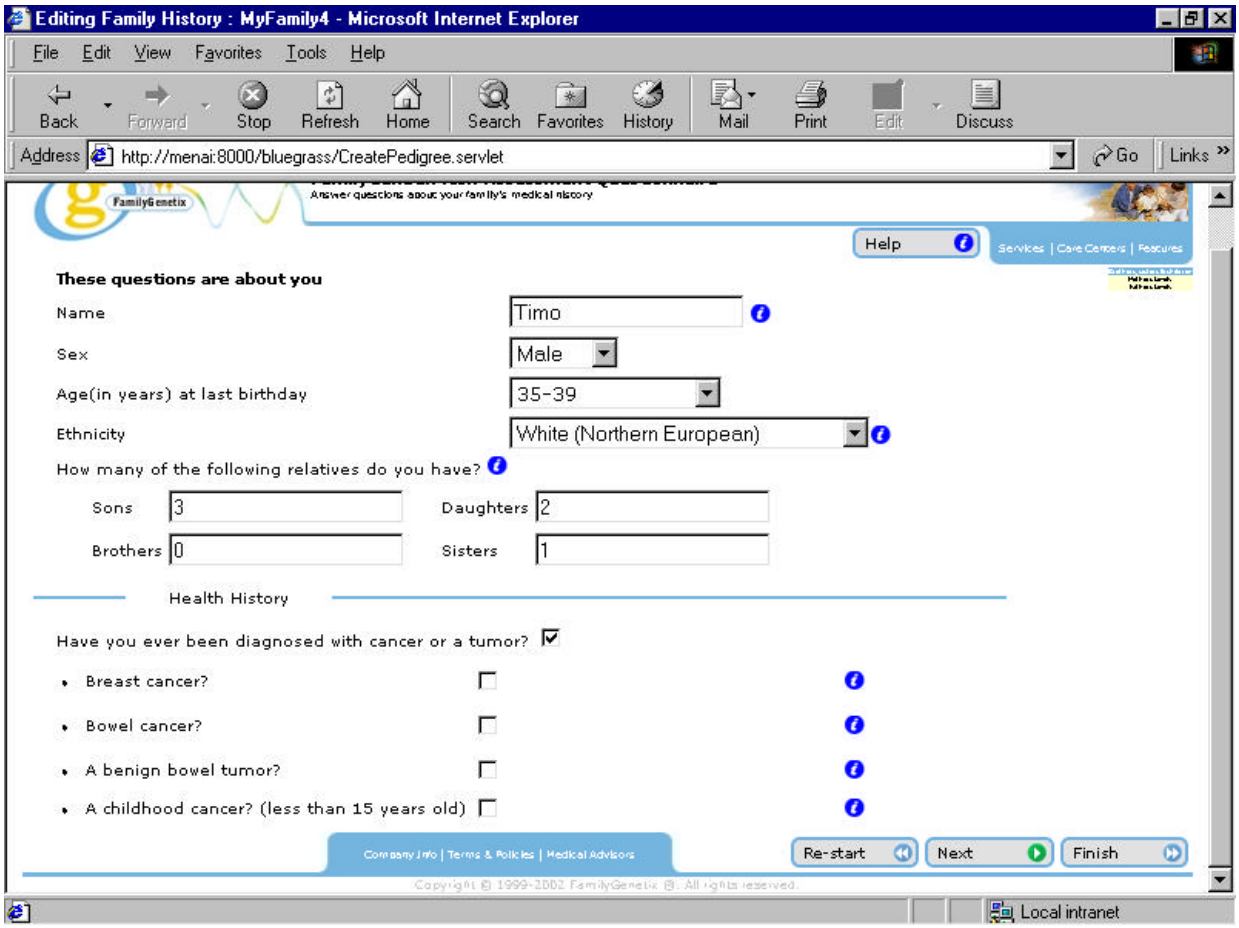


Figure 4. Window displayed after checking the 'cancer' box

In many cases, the actual value of an attribute does not affect the result of an event. For example, in this system, the only thing that triggers the display of the lower part of the window shown in Figure 4 is to check the “cancers” box. To specify s uch situations, we use the symbol '*' to indicate that an attribute can take any valid input. The following is such an example.

> [name =blank, sex =female, age =select age range, ethnicity = please select, sons =0, daughters =0, brothers =0, sisters =0, c ancers =unchecked]

{name = *, sex= *, age = *, ethnicity = *, sons= *, daughters= *, brothers = *, sisters = *, cancers =checked}

[breast cancer = unchecked, bowel cancer = unchecked, benign cancer = unchecked, childhood cancer = unchecked]

From a state transition rule which contains '*' value, a set of test cases can be derived by instantiating values entered to the control to make sure other controls will not interfere with the check box. For example, the above rule can be instantiated to obtain the foll owing rule, which directly corresponds to a test case.

[name =blank, sex =female, age =select age range, ethnicity = please select, sons =0, daughters =0, brothers =0, sisters =0, cancers =unchecked]

{name = Timo, cancers =checked}

[breast cancer = unchecked, bowel cancer = unchecked, benign cancer = unchecked, childhood cancer = unchecked]

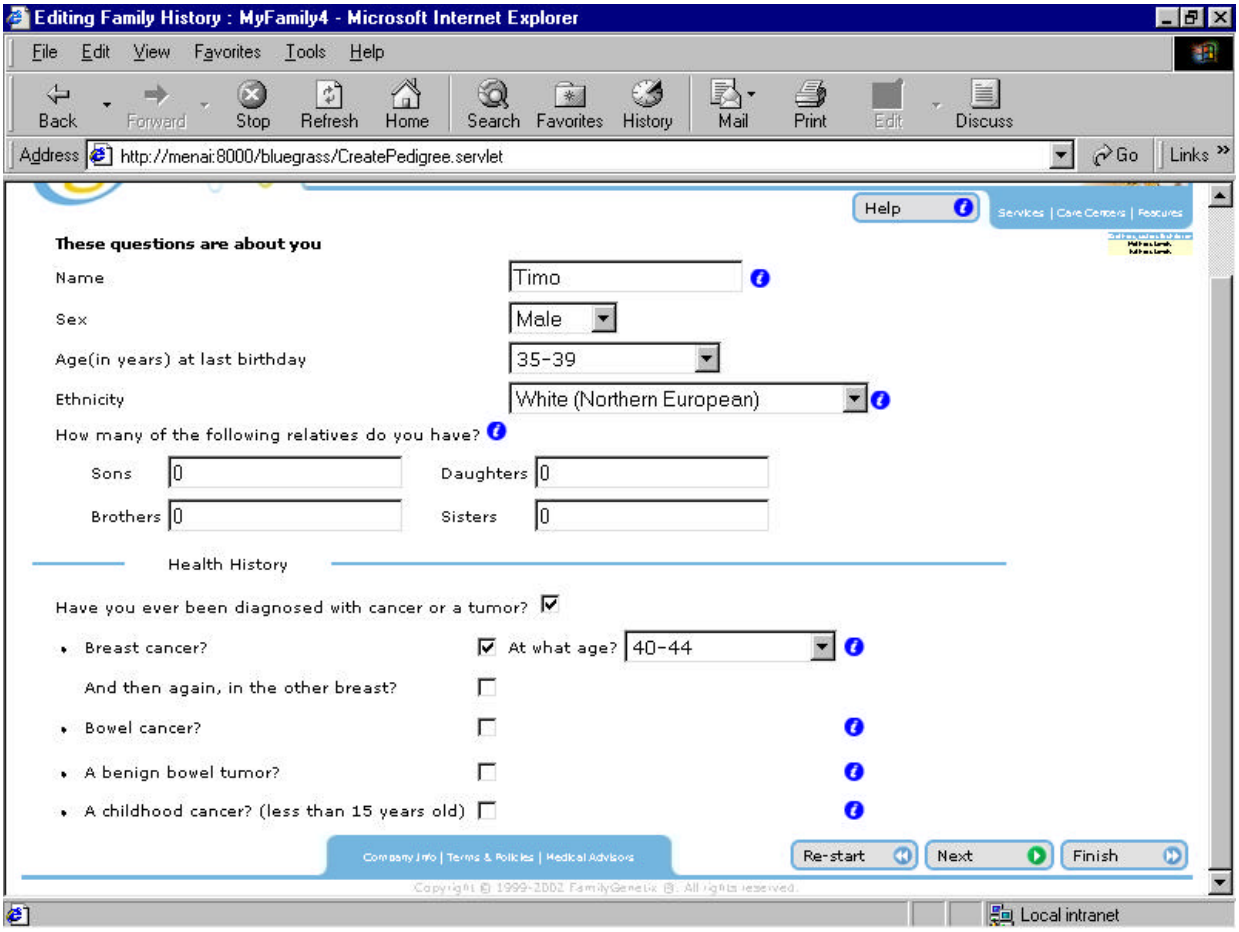


Figure 5. Window containing invalid input

The following is an example of a transition rule that displays an error message when an invalid input is entered, i.e. the number of sons is -3.

> [name =blank, sex =female, age =select age range, ethnicity = please select, sons =0, daughters =0, brothers =0, sisters =0, cancers =unchecked]
>
> {name = *, sex= *, age = *, sons=-3, brothers = *, sisters =*, press (Next)}
>
> [msgbox ("Negative sons?")]

Another example of test case that involves the testing of input validity and error message display is give below. It specifies the situation that, if the age of a disease diagnosed is greater than his/her current age, an error message should be shown. See Figure 5.

> [name =Timo, sex =male, *age =35-39*, ethnicity = White (Northern European), sons =0, daughters =0, brothers =0, sisters =0, cancers =checked, breast cancer = unchecked, bowel cancer = unchecked, benign cancer = unchecked, childh ood cancer = unchecked]
>
> {breast cancer = checked, *age-for-breast-cancer = 40-44*, press (Next)}
>
> [msgbox ("Manifestation age is greater than current age"]

This representation of human computer interactions enables testers to work on an abstract description of system's behaviors and functions in the form of GUI state changes rather than on natural-language GUI definition documents. The pre-condition of a rule that defines the state before an operation corresponds to the initial state where the tester should start. The part of the rule that specifies the event to be performed corresponds to what information the tester should enter, and what interface operation he should make such as which check boxes and/or buttons he should click and what data he should enter through the keyboard. The post-condition of a rule specifies what the tester should expect the system to respond, so it specifies the kind of verification points to be done for the interface. Another benefit of using this representation is that test cases actually form test scripts, which can be executed manually and corresponds very well to automated scripts. It can be used in practice as a readable description of what record-playback scripts do.

3.3.2. Functional testing using task analysis to generate scenarios

The technique described in section 3.3.1 serves the purpose of testing system's state transitions triggered by a human computer interaction event. For the purpose of functional testing, a systematic way is needed to analyze system's behavior to verify whether the system can perform required business functions.

There are several ways to formally derive functional test cases, e.g., from activity diagrams in UML [5]. In a real world, a tester may not have such diagrams from development team and he may not be trained to use UML. The method used here is inspired by task analysis techniques [6] in Human-Computer Interaction (HCI) research. Tasks are about behaviors. The basic idea of this method is to analyze the behavior of the specified system on a set of task scenarios. Each task scenario represents a set of situations that may occur in the use of the software system. The result of such

behavior analysis is an activity list, a notion from HCI task analysis techniques but extended to describe system's behavior. An activity list is a linear sequence of the events that happen inside the system in temporal order. Such events include: computations performed by the software; information exchanges between software components as well as between the system and its environment; changes of the system's internal states; and system's reactions to stimuli from its environment. Each action list can be used as a scenario for functional testing and can also be used as the basis for record-playback scripts.

The process of generati ng test cases using this technique consists of the following activities:

1) The selection or generation of a set of task scenarios

2) The analysis of the system's behavior on each task scenario to get an activity list

3) Executing the test

The identification of task scenarios starts with the functional requirements of the system. For example, the following are among the list of task scenarios. Each scenario corresponds to one general function of the system.

- Risk assessment for breast cancer
- Risk assessment for ovarian cancer
- Risk assessment for bowel cancer
- Risk assessment for hypercholesterolemia
- Edit family pedigree information

Each scenario is then further decomposed into a number of scenarios according to the design of the system. For our system, to assess the risk of each type of disease, the system is designed and implemented based on the sets of guidelines developed in medical professional community. For example, one of the US guidelines for breast cancer considers that the family history of a person suggests that he/she has an increased risk of developing breast cancer if he/she has 3 or more affected first or second degree relatives on the same side of the family, regardless of age at diagnosis. A scenario can be identified for the situations when th e guideline applies to a person.

However, such a scenario is still not detailed enough to test the correctness of implementing the guideline. Therefore, further decomposition is done by identifying the parameters in the guideline. In this case, the parameters include the number of affected relatives, the degree of the relatives, and the side of the family. Having identified the parameters, classic domain partition and boundary analysis techniques or decision condition testing techniques can be applied to test whether the guideline has been properly implemented. Each abstract test case represented in the form of the values of the parameters becomes a task scenario. The following are among the scenarios that used for testing this guideline.

- 3 affected first degree relatives on the paternal side
- 3 affected first degree relatives on the maternal side
- 3 affected second degree relatives on the paternal side

- 3 affected second degree relatives on the maternal side
- 1 affected first degree relative and 2 affected second degree relatives on maternal side
- 1 affected first degree relative and 2 affected second degree relatives on paternal side
- 2 affected first degree relatives and 1 affected second degree relative on maternal side
- 2 affected first degree relatives and 1 affected second degree relative on paternal side
- 4 affected first degree relatives on paternal side
- 4 affected first degree relatives on maternal side
- 2 affected first degree relatives and 2 affected second degree relatives on paternal side
- 2 affected first degree relatives and 2 affected second degree relatives on maternal side
- only 2 affected first degree relatives on maternal side ……

Such task scenarios may not be the final scenarios that can be directly used to derive activity list. However, they often form the core of test scenarios. For each of the above scenarios, a set of test cases in the form of family pedigree can be worked out. For example, for the scenario of "3 affected first degree relatives on the maternal side", we can have (mother, daughter, sister), (mother, brother, son), etc. Then, according to the process of interaction, a step-by-step activity list can be obtained, which contains a sequence of interface operation events for constructing a family pedigree and assessment. Executing of all these test cases in the form of sequences of events and comparing the results with expected outcomes, i.e., guidelines, will ensure an adequate testing of the system.

Test scenarios also come from real cases. For example, a number of real cases are collected from the uses of our previously developed intelligent pedigree management system Cyrillic. Cyrillic software can use the risk factors of family history to calculate risks of breast and ovarian cancers and also enable you to enter your own data set for inherited disorders. A large catalogue of pedigrees has been accumulated and has been used for internal testing inside the company.

3.4. Usability testing

Since our service is aimed at providing web-based tools for use by the general public, ease of use is essential. The method selected here is to record, on video, typical end-users using the software – without advice or interference [7]. This approach has been selected for the following reasons:

- It is a pragmatic measure of whether people use the software successfully.
- It requires little expertise on the part of the person conducting the test (either to help the end-user verbalize his thoughts or to aid the end-user without invalidating the test results).
- It provides an objective and complete record of each end-user's experience.

2 cameras, a scan converted, a "quad" and a video recorder were setup as shown in Figure 6, with a

partition between the computer used by the participant and the equipment used by the test conductor. A further digital camera (not shown) was used to provide a fourth image showing the whole room. There are four images on the video monitor:

- One looking at user's hands on the keyboard and mouse,
- One looking at user's face, to get your reaction,
- One showing what is on the screen and
- One showing the whole room.

Figure 6. Setup of usability testing

Ten people took part in the usability evaluation, ranging from inexperienced computer user/ not an Internet user, to experienced computer user/not an Internet user and expert computer user and Internet developer. The participant was asked to sit at computer and to put on microphone and was told that the video had started recording. Then he/she would be asked about how familiar he/she is with computer, Internet and healthcare. After that, some exercises were given to enter a specific family pedigree to assess risk. The participant can stop the test at any time. Verbal comments can

be made at any time during the test and recorded by the tape recorder. The participants were encouraged to work on the exercises without help or comment from the test conductor as far as possible. However, the conductor would answer questions when a participant really needs help. There is no time limit for each exercise. After each exercise, the following questions were a sked to the user:

- Please tell me the risk level the person was judged to be at.
- What should that person do next?
- Overall how easy did you find the software to use?
- Were the instructions clear?
- What did you think of the on-line help?
- Are there any points you would particularly like to make to us about the software or about this test?
- The participant would be asked whether he/she would like to try a further exercise, if appropriate.

Nearly everyone used the tool successfully and said it was easy to use. Takin g roughly 10 to 15 minutes to enter a family of 8 people, on first use of the tool (without any help). Several participants found some parts of the GUI (family tree review, tabs for the pedigree editor, the search facility for selecting a previously used family history) confusing. It didn't stop them using the tool successfully. It was not obvious to some users that the risk report had several pages and that they saw only the summary page. Several participants commented that it would be better to have yes/n o options rather than a check box and that they would prefer to say explicitly when information was not available.

Reviewing sessions afterwards is easy to do and works quite well. Audio monitoring is extremely useful for reviewing. The results of usabili ty testing have been incorporated into user interface design document for next release and such testing will be used at several stages in future projects.

4. Conclusion

In this paper, the experience of introducing a quality assurance system into a small comp any is reported. In the process of developing and testing web -based applications, a set of testing methods was developed.

In [8], three levels of requirements for testing hypertext applications have been identified. They are:

• the validation of the information contained in each node of the hypertext;

• the verification of the correctness of the implementation of the links between the nodes;

• the evaluation of the system structure for testing usability.

Three sets of adequacy criteria have been proposed to adequately test hypertext applications. These principles can be easily extended to web testing if each window is treated as a node and buttons caused window switch as links. However, such links in web-based applications are more complicated. Existing automatic testing tool can only partially support the testing at the lower levels. For example, a tool like Rational SiteCheck can be used to find broken links in your help. However, this cannot ensure that a link will bring you to the right place. Manually checking this for each build is labor-intensive and error-prone. It is better to be done by automated scripts guided by adequacy criteria listed above. Some verification points can be set to check things like the title of each help window. In this paper, the technology of formal representation of GUI intensive applications is developed. State transitions formally specified by pre/post conditions of interface events can be directly used to generate test cases for testing at the first level and second level. Such test cases can be easily translated into test scripts used by automated testing tools.

Due to the complexity of web-based applications, a new level of testing must be added on top of the second level for testing the correctness of the implementation of the functions. We further developed the task analysis method proposed in [6] for this testing purpose. The method has proven to be effective and efficient for practical applications.

The usability testing is of course at the highest level. However, it is essential for web sites that provide public access. The testing process can be regarded as testing whether the design of human computer interface can naturally lead the user from a given scenario to complete a sequence of interface operation activities.

It is the effort of everybody in the company to make the change happen. We have seen the benefits of the introduction of the development process and quality assurance system suitable for the scale of the company. We will further improve them to raise the quality standard of our software products and also increase productivity of the company.

References

[1] E. Dustin, et al, Automated Software Testing, Addison-Wesley, 1999.

[2] S. McConnell, Rapid Development, Microsoft Press, 1996.

[3] W. Perry, Effective Methods for Software Testing, Wiley, 1999.

[4] D. Mosley, Client-Server Software Testing on the Desktop and the WEB, Prentice Hall PTR, 2000.

[5] R. Binder, Testing Object-Oriented Systems, Addison-Wesley, 2000

[6] H. Zhu, L. Jin, & D. Diaper, Application of task analysis to testing software requirements, Proc. of SEKE'99, June 17~19, 1999, Kaisersluatern, Germany.

[7] P. Sweetenham, Usability Testing Report for GA2, FamilyGenetix internal report, Sept.2000.

[8] L. Jin, H. Zhu & P. Hall, Adequate Testing of Hypertext Applications, Journal of Information and Software Technology, UK, V39, No.4, 1997.4.

QWE2000 Session 1M

Mr. Ton Dekkers [Netherlands]
(IQUIP Informatica B.V.)

"Quality Tailor-Made (QTM)"

Key Points

- Achieve "design to benefit" by optimising the product
- Functional requirements and quality aspects
- Controlling risk on product and process

Presentation Abstract

A project runs a lot of risks related to the product to be delivered and related to the process to realise the product. Based on this you can split up risks in two groups of risks: product and process. Each group can be divided into two areas.

When you analyse the mechanism the projectmanager uses to handle the risks, five steps can be recognised. These steps are valid for both product and process.

In the presentation (and paper) the accent will be on the product. The Functional requirements are an image of the business needs and priorities. Quality is strong related to the human factor of an information system. Expectations related to the way the information system should work are difficult to manage. Both issues are very important if you want an information system "Fit for Purpose".

The other reason to address the product: on the areas planning & control and environmental factors there is a lot of literature and a lot of experience.

To handle the quality aspects in a project is one of the services of Software Control Kwaliteitszorg (Software Control Quality Assurance). SCK is looking for a practical solution to control especially the risks on the quality aspects. With Quality Tailor-Made (in Dutch: Kwaliteit op Maat) SCK succeeded in finding a solution. In the approach of QTM the match of the four risk areas with the five steps models the structure of the RADAR Method. RADAR (Dutch: Resultaatgericht Automatiseren Door Anticiperen op RisicoËs) translated in English: Result-driven Automation by Anticipating on Risks.

To identify the possible goals the quality aspects definitions in ISO 9126 are used. By using a questionnaire with a subset of questions for each roles participants can act in the project, the quality aspects identified are valued. The results of this exercise will be presented in a RADAR diagram. The plot helps in discussions and supports the specification of the requirements.

To determine the risks and to find the right measures to eliminate the risks the requirements (column) will be matched with the products (row) in the Product Risk Matrix.

This will help to identify the spots where the requirements have impact on the products and where products should be checked related to the requirements. In addition to prioritising the needs it helps you also finding the moments to check whether you match the expectations. The earlier the check is, the better the product will fit to purpose: "design to benefit".

RADAR guides you through the five steps to handle the quality aspects in a way that it fits the project: Quality Tailor-Made.

About the Speaker

Ton Dekkers has been working as a practitioner and manager within the area of software quality for a great number of years. Within this area he specialises in estimating, risk analysis and QA in projects (QTM in practise). He is a regular speaker both at national and international conferences and a trainer in software estimating, risk management and QTM.

Ton Dekkers is quality consultant in the division Software Control Kwaliteitszorg of IQUIP Informatica B.V. IQUIP is a Dutch information service organisation (1450 employees) that is mainly active in building, maintaining and implementing information systems. The process is supported by services from the Software Control divisions Kwaliteitszorg (Quality Assurance) and Testen (Testing). He is member of the NESMA workgroups FPA in Maintenance and FPA and New Technology.

Quality Tailor-Made

Ton Dekkers

Abstract

"We boldly go where no man has gone before". We want to announce new products and services and use matching supporting information systems. All this more rapidly than before and with higher quality. Rapidity and quality seem to be conflicting but that is not necessary. If we want to identify and trace fast moving objects we use radar. In this paper we introduce a new RADAR. A method that makes it possible to manage rapid developments adequately by identifying and eliminating the unwanted elements in time. By this we achieve Quality Tailor-Made (QTM).

Introduction

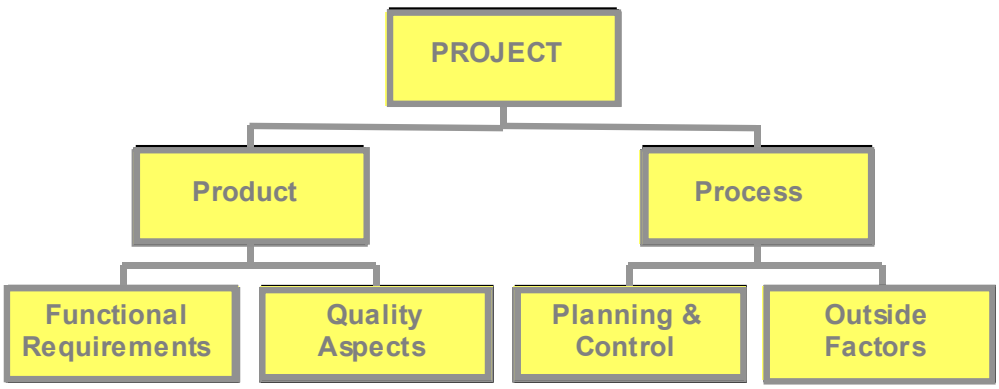
To lead a project to the ultimate goal, all threats to achieving this object must be identified and eliminated in time. Who does not have experiences like:

- under pressure of time, towards the end of the project, functionality will be "stripped";
- in a project with a fixed end date, control is focused on the deadline;
- in a project with a limited budget, control is focused on costs;
- changes or reorganisations have a direct impact on the project organisation.

Software Control (QA) has looked for an approach for managing all the above mentioned aspects, both prior to and during the project. In this paper no new development method is presented: the existing methods provide solutions for all conceivable problems when building an information system. The message of QTM is: do not consider automation projects from a point of view of time or money, but choose an approach that takes the risks into account. What threatens the project? What can go wrong? Virtually everything in terms of achieving the goals of the project within the constraints of time and money.

Risk areas

The risks a project runs can be split into risks related to the deliverable product and risks related to the process to get to the product.



Regarding the product, two kinds of risks: risks can be identified: risks pertaining to functionality and risks related to quality aspects. The first group concerns risks regarding the functions that the information system offers the user. Under quality aspects we have to think

of all "other attributes" of an information system, like the processing speed and security of the system.

On the process side two types of risks can be distinguished: risks in relation to the internal planning and control of the project and those in relation to the outside factors.

Briefly, the risk in respect of the product is the right system is not built and the risk in respect of the process is: the system is not built right.

The task of the project leader is to cover all four risk areas; all possible risks must be recognised and monitored continuously. The project leader must constantly pay attention to all that comes within his focus area. In aviation and shipping it is also necessary to know what takes place in the controlled area. For this for they make use of radar. The radar has a number of functions:

- The radar identifies moving objects. With the radar every moving object that enters the controlled area is detected.
- Next the radar investigates what kind of object it really is: a known or an unknown object? And what is the course of the object: is it dangerous?
- In defence applications the radar data can be passed on to the defence system, which eliminates hostile objects. In public applications data can be passed on to a warning system.

In principal the project leader does the same as this radar, but in relation to the risks: look what is coming towards you, identify the risks and take the appropriate measures timely. These measures must “be available”, they should be defined beforehand. When we consider the "radar" of the project leader in detail, five steps can be recognised. These steps are aimed at eliminating the risks regarding the product and pertaining to the product development. These five steps will be considered for all four risk areas: Functional Requirements, Quality Aspects, Planning & Control and Outside factors.

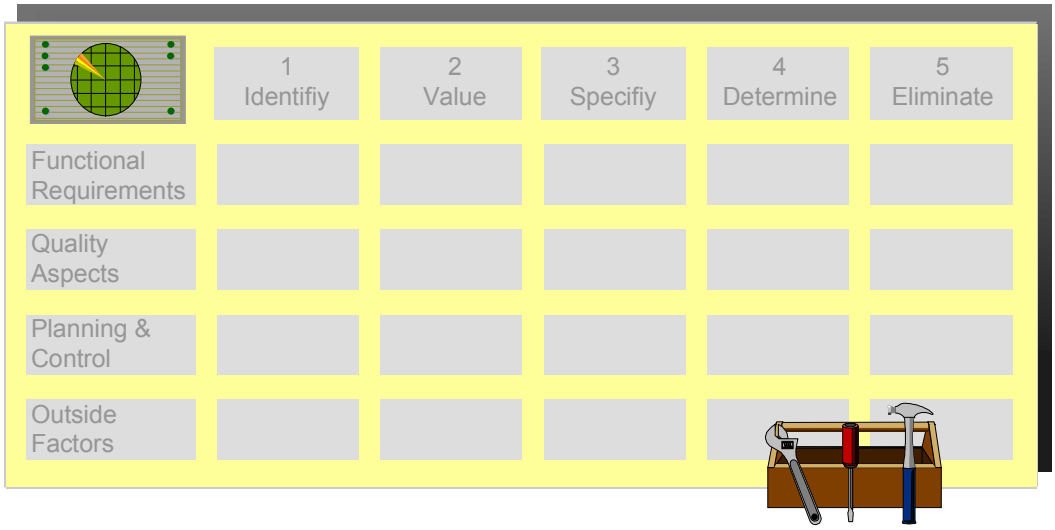
The first step is to *identify*: what do we really want? In this step we will try to identify all possible aspects within the four risk areas respectively. When we *value* the objectives, a selection will be made from all aspects that are relevant for this specific project. There is always a hierarchy in the properties. A "short list" is made based on value. From properties which are considered to be important, the requirements must be *specified* in detail. For instance, if response time of the system is important, then criteria for response time must be specified and quantitative aspects like the percentage tolerance must be defined.

There are still two more steps to go. Once the requirements have been made explicit, we have to *determine* which risks there are. And finally the risks must be *eliminated*: measures should be taken.

Summary: in five steps the needs will be defined, valued and worked out in demands and wishes, after which the risks will be mapped and measures taken to cover these risks.

QTM matrix

The QTM matrix is created when the four risk areas (Functional Requirements, Quality Aspects, Planning & Control and Outside factors) are combined with the five steps (Identify, Value, Specify, Determine and Eliminate):



For a project that is using this method, each cell of the matrix contains specified actions. However it is possible that a cell remains empty; in that case there is a well thought out reason: you consciously choose not to take action and you are aware that you take a risk.

Traditional development methods are mainly based on time and money: they are all focused on controlling the process in terms of time and budget. Of course, in these methods Functional Requirements and Quality Aspects are also important, but the main goal is still: finish the project within time and budget. The quality of the deliverables comes last. In practice the system delivered (on time) often does not offer what the user expects. The system does not meet the expected quality.

RADAR is a practical method based on quality. The user must accept the information system that will be built and the user must be able to work well with the system. In this approach time and money are “deliverables”. If these outcomes do not meet the requirements imposed by constraints of time and money set by the principal, one need to consider the relevance of and balance between functionality and quality. At the end all should be in balance. The result is an optimal mix of functionality and quality serving the business within the constraints of time and money.

RADAR stands for "Resultaatgericht Automatiseren Door Anticiperen op Risico's"(translation: Result driven Automation By Anticipating on Risks). This is the crux of the method. Assure that the demands and wishes are well mapped, determine the risks and the consequences of those risks. When you know the consequences, you can decide how much time and money you want to spend to eliminate or control the effects of the risks.

RADAR is a preventative approach and can be used within existing system development methods like SDM, RAD/JAD, etc. Risks always occur no matter which development method is used. The RADAR-method helps by assessing the risks and the consequences. RADAR not only considers the process of the project, but also pays attention to the product and the risks related to the deliverables. In this paper we look specifically at the product side, i.e. the Functional Requirements and the Quality Aspects whereas the process side is only dealt with briefly.

Functional Requirements

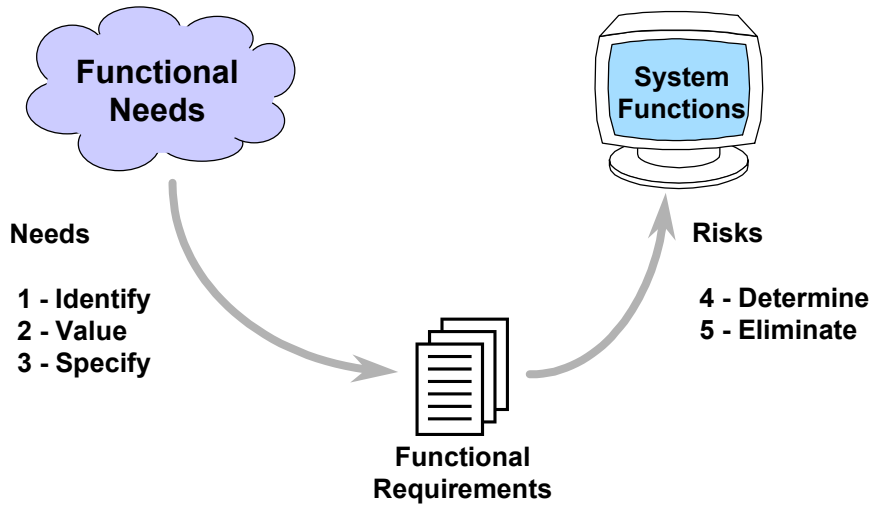
The Functional Requirements comprise all that the information system should offer in order to support the business process. *How can we prevent these problems? How do we get from the thoughts (needs) of the user to requirements of the information system?*

First we must translate all that is in the minds of the users into concrete instructions for the developers. *What does the user want?* The conversion of the functional needs into demands takes place in three stages:

- Identify functionality: we make a list with functions together (all participants).
- Value functionality: which functions are more important and which are less important?
- Specify functionality: the user himself puts on paper what he expects from a function.

Then these demands are converted into the system functions of the information system. *How does the developer realise the requirements that the user has specified?* To control this translation on needs to:

- Determine risks: what risks does the developer run when building the system?
- Eliminate risks: take measures to avoid the risks or to limit damage as much as possible.



QTM supports these steps with a characteristic approach.

- *step 1: Identify functionality*
  The needs of the various users will be identified in group sessions. The result of these sessions is a total vision that represents the view of all of the users.
- *step 2: Value functionality*
  The importance of the functions will be valued from the point of view of:
  - the business: how important is the function for the business process?
  - the user: which functions are important for operational tasks?
  - the administrator: which functions are necessary to manage and maintain the system?

- *step 3: Specify functionality*
  The user(s) specifies the main functional aspects so the developer can form a good picture of the system that should be built. In group sessions the requirements are drafted and checked on feasibility (technical and budget).
- *step 4/5: Determine risks and Eliminate risks*
  After the specifications are drafted, the (standard) risks are considered: the consistency of the requirements (several translations depending on the development method) and quality of the data (reliability, actuality and conversion).

Identify Functionality

Within QTM the requirements will be established in group sessions, rather than only using interviews with the individual parties concerned. This has a number of advantages:

- Parties concerned see and hear about each other's problems and interests. They understand the relation between and consequences of decisions made.
- The participants do not need to read large documents to find the results of the interviews.
- All stakeholders participate in the group sessions. It is less time consuming because the tuning of the requirements is done in the same sessions.
- The participants identify the needs. They do not need to agree upon requirements identified by a "designer" based on interviews. And the needs are written in a manner that is comprehensible for participants.

It is important to start with clearly defining the scope of the information system. Although it is considered "not done" to inquire about the constraints of time and money, this information is crucial. If size of the budget cannot be defined, then it is not possible to make a statement about the permitted size of the requested information system. Functional size metrics, like function points [7][8], help to set the correct boundaries and to limit the identification of the needs. Based on these figures we set up requirements tailor-made. The development process should provide the appropriate metrics. More about metrics later in this paper.

The stakeholders (business, user and administrator) create in the group sessions a list of the functions they need to do their job. The business requirements needs to be full filled by these functions. Other participants, like operations and control, add the functions needed from their perspective to complete the system. In a special session (the consolidation session), all participants agree upon the list: "This is the system that we need".

Value Functionality

Before we start to value the functions that have been identified, we need to have an idea of the functional size of the information system. The discussion will be influenced by the fit of the set of requirements within the budget. In practice there are usually more requirements than we can handle given the stakeholders' time and budget constraints. By using functional sizing metrics we can estimate the size of the functionality. Based on benchmarks (internal or external) the effort for the development of the information system can be calculated. This will help the discussion about necessity, importance, priority and value for money.

Valuing the list of functions that is composed in step 1, is done in the same way as identifying the functions. In group sessions with the relevant stakeholders the functions will be discussed. Necessity and importance are the aspects to weigh the functions. Based on this discussion stakeholders prioritise the requirements and decide which requirements will

actually be developed within this project. The other functions can be scheduled in further releases.

A technique that can be used to investigate candidate requirements is the Analytic Hierarchy Process [3]. AHP compares requirements pairwise according to their relative value and impact. This approach includes redundancy and is thus less sensitive to judgmental mistakes.

The functions to be developed can be classified on priority. When we also classify on order to develop, the stakeholder will first get the functions that he needs the most. At the same time we minimise the risk to develop an incomplete, not working system when the budget is reduced.

Specify Functionality

The requirements should be specified by the user. Tasks and activities of the user are leading. The activity brings on a certain result. To get to that result the user needs input. In principle the requirements should reflect the tasks and activities of the user getting from input to result. That's why the user must be actively involved. We use again group sessions to achieve optimal participation. It is recommended that an information analyst should be present in the sessions to check the feasibility of the specifications. He or she could advice alternative solutions when applicable. The alternative could be a cheaper or more effective way of performing the tasks.

After this step the needs are specified in the requirements. In the development process the requirements will be transformed to the functional design.

Determine Risks and Eliminate Risks

When the specifications are ready, the risks are considered. Standard risks are the consistency of the requirements and the quality of the data.
In the process of transforming the business requirements into an information system several translations are made. How can we guarantee the build system does reflect the requirements?

- Using a system development method;
- By performing consistency checks e.g. (Fagan) Inspection, review, …

The quality of the data is perhaps the most important. Even with a correct information system when data is poor, the output (result) is poor: garbage in = garbage out. Measures to take care of the reliability:

- Implement a continuous check on the accurateness of the data;
- Remove redundancy of data;
- Limit the update points and the persons that can make updates;
- Define a conversion strategy, make a conversion plan;

Project

Although the requirements are valued and clearly specified, there still are risks in developing the system. Risks like the order of development of the functions, did we miss essential functions or did we choose expensive solutions while we have a limited budget.

QTM includes the Product Risk Matrix to determine an eliminate risks. This Product Risk Matrix is also used for risks related to quality aspects and will be explained later.

Quality Aspects

When talking about Quality Aspects all "other" properties of the information system are included. Aspects that can be seen as the Human factors of the system.

When developing an information system a number of problems arise. The project team has to deal with users who have their own picture about what the information system should do.

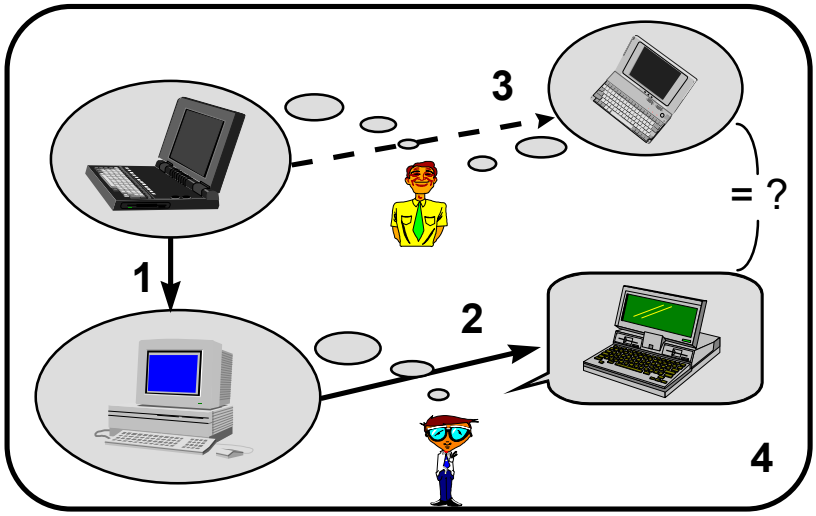
The first challenge for the project team is to get the correct picture, matching all views. We used group sessions to define functional requirements that are accepted by all. For quality aspect we need another approach. Quality aspects are related to the way the system will perform and will decide the outlook of the system.

Once it is clear what the system should do, the system must still be built (second challenge). During development there are many reasons why the system developed may deviate from the specified system.

A third challenge is the changing expectations of the user. While the project team is developing the system, the user creates new ideas and requirements, so he adjusts his expectations. In this case the project team delivers exactly what was wanted originally but still does not fit with the (new) expectation of the user. When it turns out that expectations change during the project:

- The project will be adjusted, to meet the new expectations. Undoubted with consequences for time and money.
- The expectations will be adjusted and set back to the original expectations. In this case no additional budget is required.

Changes in the project organisation can also cause variations of the expectations. In particular when people in the project will be replaced. New people means new ideas about functionality and quality aspects.

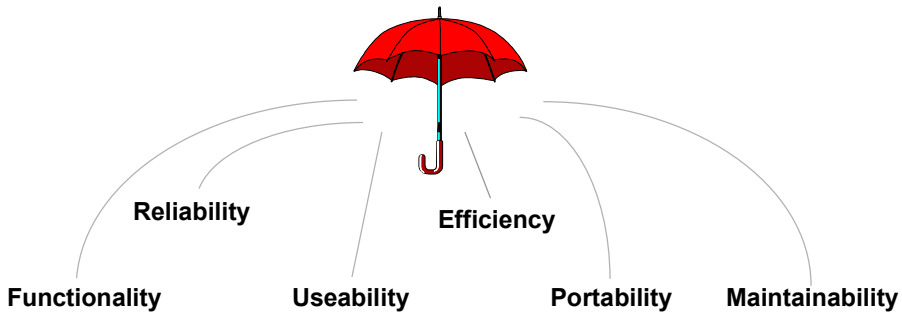


The RADAR method supports the solving of these problems, related to functionality as well as to quality aspects. Quality aspects is the collective noun for all nonfunctional properties that make up the success of a system. Properties like performance and security are typical issues that belong to the group of quality aspects.

For quality aspects the five steps of RADAR also apply: identify, value, specify, determine and eliminate.

Step 1: Identify

An international standard (ISO-9126 [4]) is available to identify the quality attributes of an information system. This standard is used as starting point. In the project Quint II [5], the ISO attributes are expanded and some attributes are added, resulting in the Extended ISO model. Under an "umbrella" structure the attributes are clustered in six main areas.

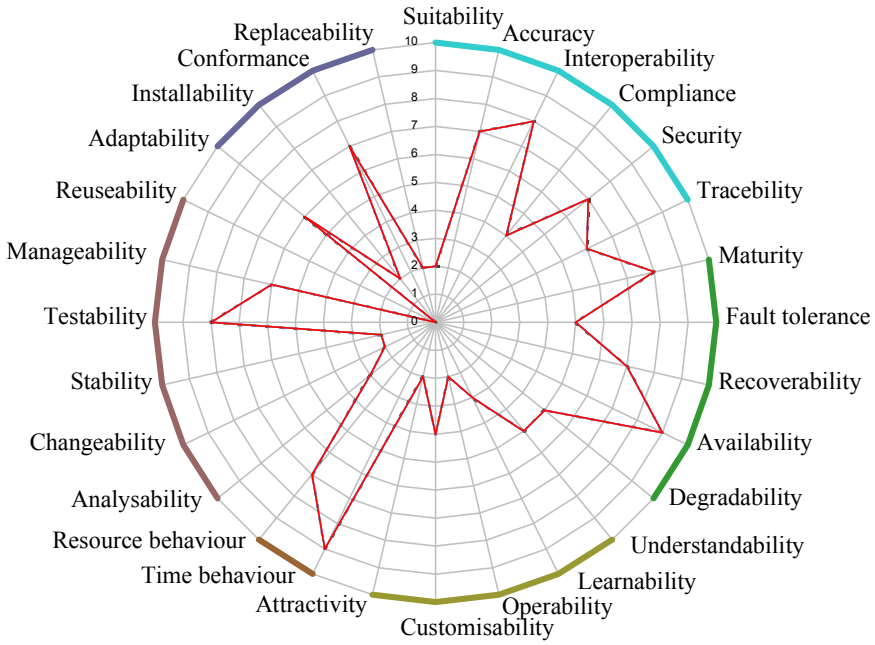


Step 2: Value

When valuing, all attributes from the list (Extended ISO) that are relevant for the required information system will be investigated. We use the same point of view that is used when checking Functional Requirements: Business Importance, User Importance and Administrator Importance. Each party has a different view of the system and has different (sometimes conflicting) interests. Sometimes, there are even more parties that have specific requirements e.g. Accountancy and Security. The number of "customers" of the system extends, and each "customer" expects to get what he wants and what he needs to do his job.

The biggest problem is communication. It is difficult for the user to indicate on a list of quality attributes what the important ones are: everything is important and choosing is impossible. In short, choices about properties and valuing are not so easy.

QTM uses the Quality Radar to value the properties. The user always gets two propositions from which he must choose. Each proposition is linked to a quality attribute. By making a choice with the required information system in mind, the user indicates the importance of the "other" properties of the system. The result is a Quality Radar:



On the circle you find the quality attributes and the plot indicates the importance of the properties for the user who completed the Quality Radar. This tool enables simple comparison of the opinions of various users. The graph does not tell what the system should look like or do, but:

- The users have to think about the properties.
- By talking about the system: the quality of the system will be discussed, as well as the reason(s) why some of the quality attributes are more important than other quality attributes.
- The Quality Radar is a powerful communication tool: it gives a good basis to discuss the system and leads to greater involvement of users and other stakeholders.
- Communicating with the help of the Quality Radar leads to agreement between the users about the (most) important properties of the information system.

A participant chooses a point of view (business, user or administrator). Each fills up results in a Quality Radar. On the basis of various Quality Radar's, all participants identify the most important properties of the information system.

In QTM we use QUINT II as reference model but the same method can be used for ISO – 9126 [4], the quality attributes of Tmap® [6] or a company defined list.

Step 3: Specify

The (most) important properties must by specified. If "Availability" is considered to be important, the user must define concrete requirements, e.g.: "the information system must be operational five days per week, eight hours a day". The properties must be defined in such a manner that during the project the system can be checked against these requirements. By "Operability" we mean the users want an easy to handle information system. This requires for example uniform screen layouts, not only within the system, but also across other existing systems.

Step 4 and 5: Determine and Eliminate

When the properties are known in detail, the developer has a good understanding of what should be built. Further, he has to think about: what kind of risks do we run in developing the system? If considering "processing speed" (time behaviour), a concrete requirement is for example that a certain transaction should be completed within two seconds. Is it easy to accomplish, or is it a problem - due to the way the transaction is defined, the performance of the network, etc.? If it is a problem, measures must be taken to meet the specified requirements.

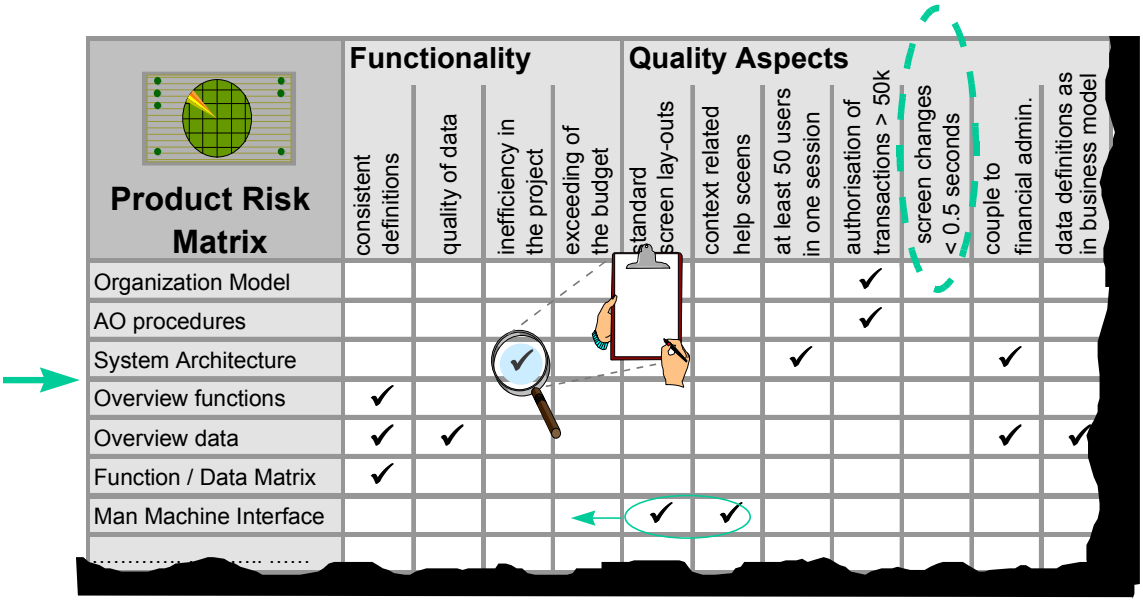
When we have determined the risks, the question remains: what are we going to do about it? How do we eliminate the risks? Various types of measures are possible:

- preventive, the occurrence of risks can be prevented;
- detective, the occurrence of risks will be located;
- corrective, possible causes will be removed and
- repressive, the consequences will be suppressed.

A possible fifth type is registrative measures, collect metrics to initiate future process improvements.

Product Risk Matrix

An essential part of QTM is the direct linking of measures to products. In this way the measures can easily and concretely be defined and the results can be made visible. The RADAR method includes the Product Risk Matrix. In a matrix the columns reflect the possible product risk areas (Functional Requirements and Quality Aspects) and the rows, chronologically represent the deliverables of the project.



The list of deliverables (products and intermediate products) is directly related to the development method used. The traditional waterfall approach gives another list than the RAD method or a package selection project. It is always possible to draw up a list of deliverables not only documents or coding; for instance a "prototype" is also a kind of intermediate product. Products like "Functional System Design" are too generic; the matrix works better when products comprise smaller logical units, such as function structure, data model, user interface and a description of the administrative organisation. Only those products that describe the information system will be included in the matrix (system documentation), not the project documentation (like plans, phase reports, etc.).

A project passes through the steps: Identify, Value, Specify and Determine. This results in a limited number remaining risk areas on the horizontal axis of the Product Risk matrix. Related to these concrete requirements the project runs risks in regard to the total quality of the product: Functional Requirements and Quality Aspects.

For these risk areas the concrete requirements are known. Next we mark (√) in the Product Risk Matrix in which (intermediate) product at the earliest point where we can test that the requirement for the (intermediate) product is met. By these marks the requirements are linked to the deliverables of the project. When filling in the matrix a problem may occur: suppose there is no single deliverable which will test the fulfilment of the specified requirement (this must be possible in the final product, at the end in the program, otherwise the requirement is not very significant for the project). A classic example is "time behaviour". Perhaps the infrastructure used gives some information, but in the end it is often not possible to demonstrate through (intermediate) results that the final performance requirements will be met.

The project leader has two options at that time:

- He accepts the risk and establishes at the end of the project whether he has achieved his goal or not;
- He does not accept the risk and takes additional measures to find out earlier in the project. The project leader adapts the list of deliverables according to the risks he has determined.

The opposite situation could also appear: if there are no marks after filling in the matrix for certain deliverables, the legitimate question is: "Should we produce that product?" The essentials are not proven? In practice it is possible the product represents a useful intermediate step towards the next product. In that case the specific product will be produced, but not maintained and will not be seen as a system deliverable. The result is a list of (intermediate) products, from which it is really clear why these products are produced.

A checklist which belongs behind the marks in the matrix where you can see " *how* to meet a specified requirement". The checklist is the basis for the check and test activities. The Product Risk Matrix is the basis for the specification of product related measures. In the matrix concrete measurable requirements are linked directly to physical system products. The project leader can take appropriate measures based on a risk analysis. The big advantage of the matrix is that all the measures are directly related to system products. The (intermediate) products will be used early in the project to ensure specified properties of the end product.

Project strategy

Ultimately QTM leads to a project quality strategy. Based on the Product Risk Matrix the project approach should be adjusted: add and/or remove deliverables. Briefly, the definitive list of deliverables will be defined, then the strategy must be defined, while the measures must be specified in a quality plan and/or a risk plan:

- Lay down product standards: requirements will be seen in at least one deliverable.
- Schedule reviews, checks and test activities;
- Decide on the techniques to be used, depending upon the products to be reviewed, the way those products will be reviewed and which techniques fit best.

Choices should be made based on the Product Risk Matrix which shows the total overview of the risks and the relation to (intermediate) products: at which moment and by whom will the checks be executed, on which (intermediate) product, and what will it cost? Decisions can be made, together with all participants on the basis of a cost / benefit analysis: do we take this measure or not? It is the responsibility of the project leader to decide if it is more efficient to combine test activities, to add additional products and so on.

Now we are sure all things are clear related to the product, we have to consider the development process. The system defined should be built.

Process Risks

To make a project successful it is also necessary to look at the quality of the process. The risk areas related to the process are: planning & control (inside factors) and outside factors.

There's a complete library available with books, reports and other documents about the development process, so what can we add to it? We said earlier QTM / RADAR is a preventative approach that can be used within existing system development methods. The chosen method influences the Product Risk Matrix. In the PRM the products (deliverables) depend on the method. Here we see a direct link between process and the ultimate product: the information system.

A process usually starts with a projectplan. In this plan the goals for the project are set. Making this plan the RADAR method is also very useful. The projectleader identifies the issues that are important for the project, e.g. development platform, organisation, schedule, resources, securityplan, qualityplan, etc. With the stakeholders the issues are valued and specified. After that the projectleader should perform a risk analysis (determine) and define the appropriate measures to attend to the risks (eliminate). At the end all of this should be reflected in the projectplan.

The most important items for stakeholders are Time and Money: when is it finished and what will it cost? We used the budget to limit the product (functional size). We could do this on account of the correlation between functional size and effort. Effort can be expressed in time or in money. Both are related to the available resources, e.g. experienced developers are more expensive than inexperienced developers but experienced developers are more effective; they do their job faster. Another main variable of effort is the development platform and the available tools. The collective noun for all effort related variables is productivity attributes.

Metrics

The planning in the projectplan is the basis for controlling the project. The planning is mostly based on experiences of the projectleader and projectteam members. Better is it to combine these experiences with objective methods like function point analysis [7][8]. By using an internal or external benchmark we can calculate the effort needed. To adjust the result we have to know the impact of this specific project on the standard values of the productivity attributes. A benchmark is based on experiences in the past. This means that projects should produces metrics to fill the benchmark. Fortunately QTM requires a limited number of metrics and productivity attributes. QTM just needs productivity figures, e.g. hours per function point / price per function point. These figures should be related to the development platform and the activities of the development method. If that is the case, then we can use these figures to determine the size of the functionality that fits within constraints of time and money.

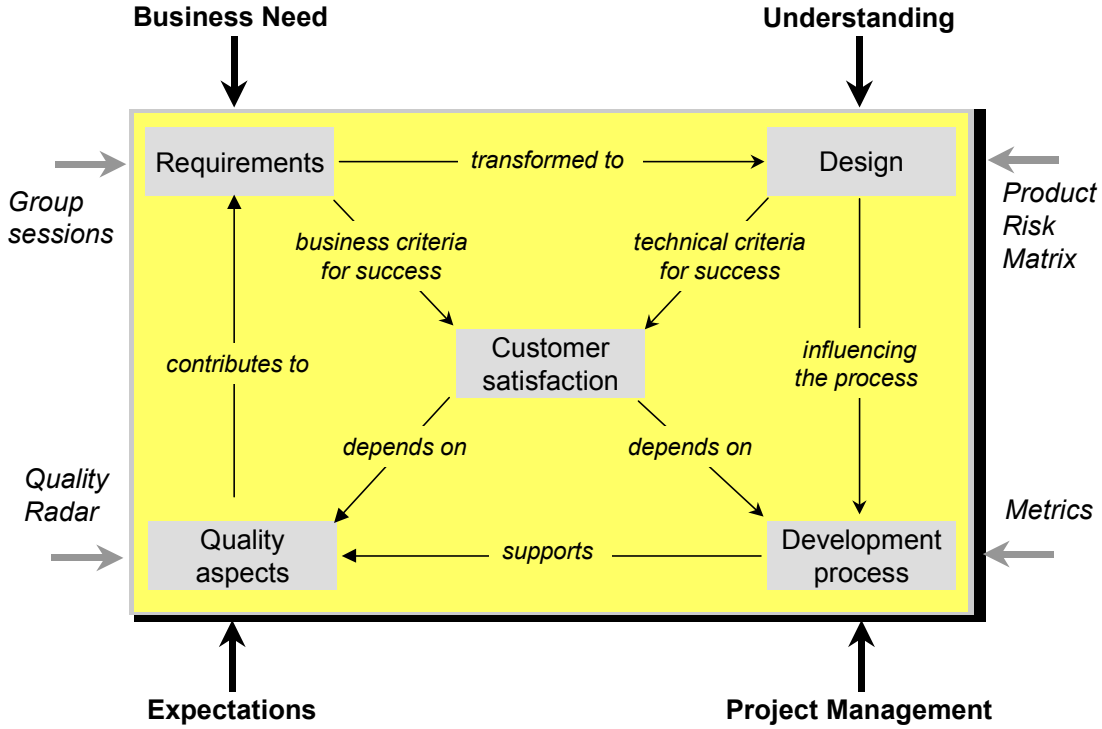
Depending on other needs we can enhance the metrics. A method that will help to set up a tailor-made metrics program is the method Goal-Question- Metric (GQM) [9].

Customer Satisfaction

The main goal of QTM is to achieve customer satisfaction. A customer reference model is used to understand from which customer satisfaction can be assessed. The model expresses the customer satisfaction using four key related components. As shown in the figure, the model components are:

- Business requirements (need for new business strategy and change).
- Functional design (transformation of the requirements including quality levels and process criteria and constraints).
- Development process (process capability and people competence).
- Quality aspects (fit for purpose).

This model represents optimal result if all its components maximise their contribution to customer satisfaction. For example, customer satisfaction is achieved when the product (information system) matches the need that is also accurately transformed and expressed in the design. The design also influences the development processes, the project organisation and competence of team members involved. The process execution will deliver trustworthy product quality.



The left triangle reflects the product area and the right triangle the process area. In both QTM components support the aim of getting customer satisfaction. For each key related component there is at least one QTM component. For example, the Requirements are drafted in group sessions. But metrics and AHP also supports that process.

The transition from product area to process area is made visible in the Product Risk Matrix.

RADAR

The RADAR method offers a structured way to go through the five steps; Identify, Value, Specify, Determine and Eliminate and offers the following advantages:

- Projects can be managed by focusing on Product as well as on Functional Requirements and on Quality Aspects. The traditional "controls" of Time and Money will be deliverables.
- The product quality will be discussible and measurable, concrete requirements are defined for all relevant properties and the meeting of requirements is tested during the project.
- It is an important tool to improve the communication. By talking about the results of the Quality Radar more is revealed than when just going through a list of attributes. At the same time the participants are confronted with others points of view.
- The project approach that is chosen and the decisions that are made can be made clear.
- Measures will be taken only when a risk is identified and the measures will be linked to deliverables with the help of the Product Risk Matrix.
- Based on the total overview from the Product Risk Matrix it is easier to make choices. You can not do everything that is important, choices have to be made.
- Using the "list" of (intermediate) products it is possible to perform an impact analysis when the circumstances of the project change. Does it cause new risks? Should there be additional test points? Should we produce an additional deliverable? With the help of the Product Risk Matrix it is easier to communicate with the owner about additional activities.
- The experiences acquired in the project lead to learning effects and improvements in the checklists, for example.

Conclusion

The approach is to define in a structured way a set of measures to manage the quality in the project. Using the Quality Radar it is easier to communicate about quality with the participants. The consequences of choices are clear and relevant.

We offer: **Quality Tailor-Made**.

References

[1] KoM, A. Boeters, B. Noorman, "Kwaliteit op Maat", Kluwer Bedrijfswetenschappen, Deventer, ISBN 90-267-2579-5.

[2] QTM, Ton Dekkers, "Project Control: The Human Factor - Quality Tailor-Made", Proceedings ESCOM-SCOPE 2000, Shaker Publishing, ISBN 90-423-0102-3.

[3] AHP, J. Karlsson, K. Ryan, "A Cost-Value Approach for Prioritizing Requirements", IEEE Software, september/october 1997.

[4] ISO 9126, Information Technology, Software quality characteristics and metrics, ISO/IEC.

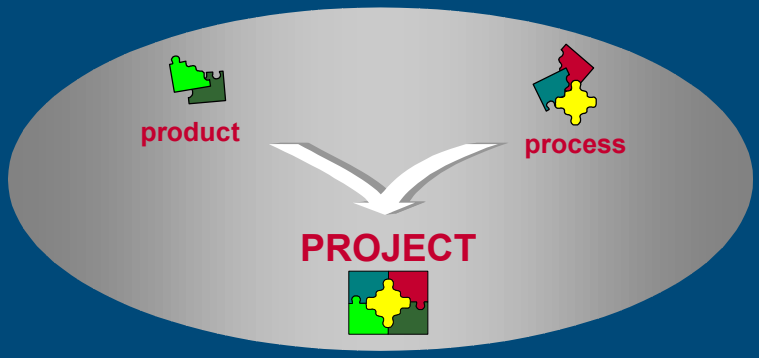
[5] Quint II, B.van Zeist, P.Hendriks, R.Paulussen en J.Trienekens, "Kwaliteit van softwareproducten", Kluwer Bedrijfswetenschappen, Deventer, ISBN 90-267-2430-6.

[6] TMap®, M.Pol, R. Teunissen, E.van Veenendaal,"Testen volgens TMap®", Tutein Nothenius, ISBN 90-72194-58-6.

[7] FPA, IFPUG (International Function Point User Group), "Function Point Counting Practices, release 4.0", IFPUG, January 1994

[8] FPA, Ton Dekkers, "Van Functiepuntanalyse naar Integrale Functiepuntanalyse", IQUIP Informatica B.V.

[9] GQM, R.Solingen, E. Berghout, "The Goal/Question/Metric method: a practical handguide, McGraw-Hill Book Company, ISBN 0-07-709453-7

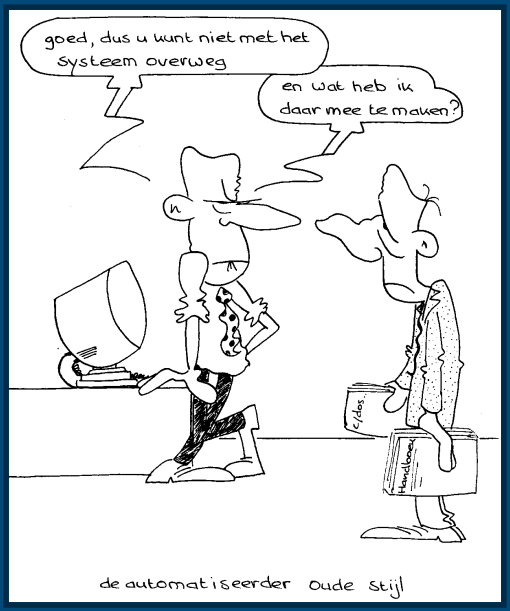


Quality Tailor-Made

A practical approach
to control project risks in IT-projects

IQUIP Informatica B.V.

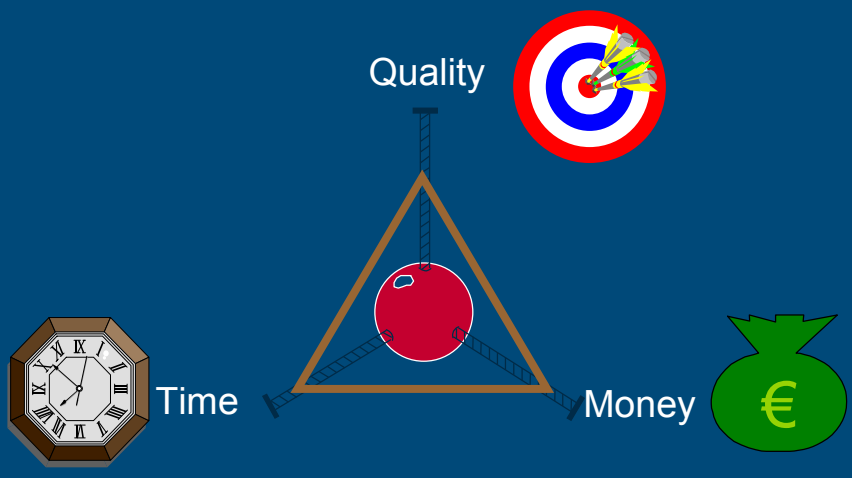
- Founded in 1972
- Full-time staff 1,500
- Cap Gemini N.V. , 100% shareholder
- Business units
  - E-Business (5 regional units)
  - Application Service Management (ASM)
  - Integration (Int)
  - Software Control (SC)
    - » Quality Assurance
    - » Testing
- Services
  - implementation of projects
  - posting of specialists



O.K., you can't get on well with the system.

Do I have something to do with that?

Object of the approach



Quality Assurance in Projects

- Review, check, test everything
- Standards to have standards
- QA sets a standard

- QA activities on 'critical path'
- High costs (time / money)

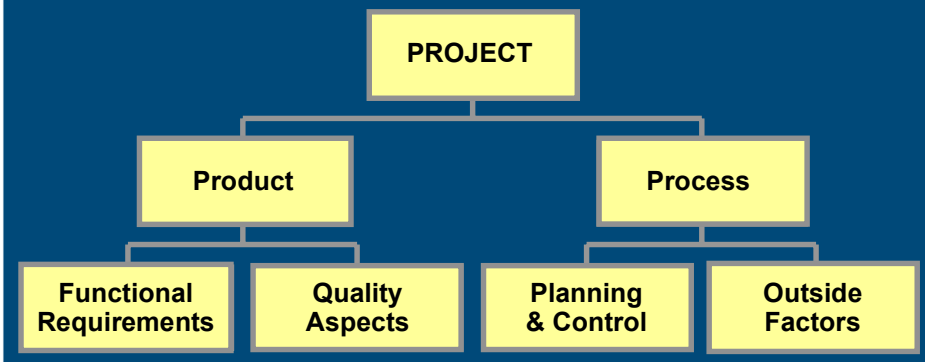
- Many (exciting?) findings
- Coverage uncertain

- Delay
- Irritated developers
- Unclear return

RADAR method

- Measures only where necessary
- Related to risks
- A clear sight for the user
- Maximising result

Four Risk Areas



Five steps to control

Map expectations:

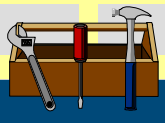
- What are the project deliverables / what are the conditions?
- In which order (priority)?
- Make expectations concrete (the project team and the user)

Take appropriate measures:

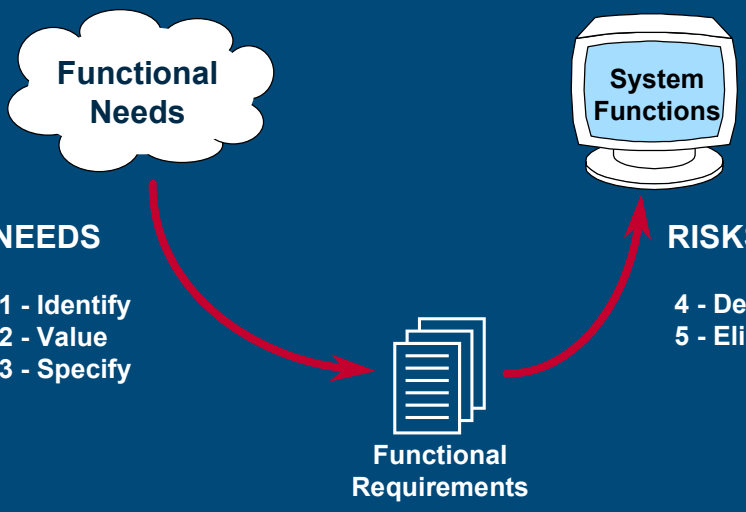
- Which risks are seen by the project team?
- Which measures are necessary?

QTM Matrix

	1 Identify	2 Value	3 Specify	4 Determine	5 Eliminate
Functional Requirements					
Quality Aspects					
Planning & Control					
Outside Factors					

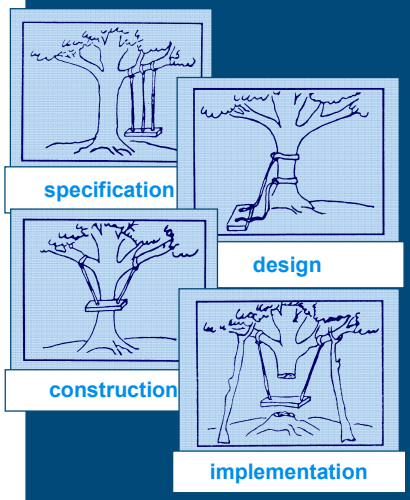


Functionality



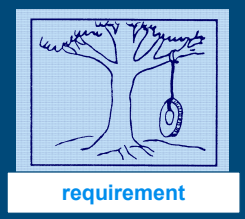
IQUIP / SCK
QWE 2000 - 9

Functional Requirements

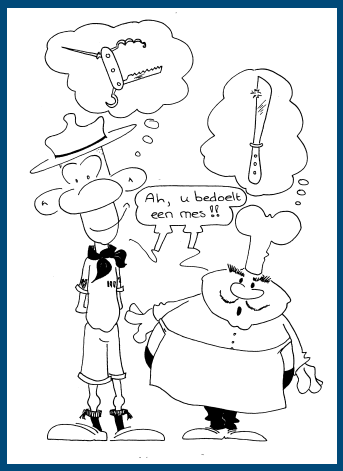


RISKS

- Fancy extra's included
- Wrong understanding
- We know what they need
- Uncontrolled requests
- 'Forgotten'
- Bugs
- ...

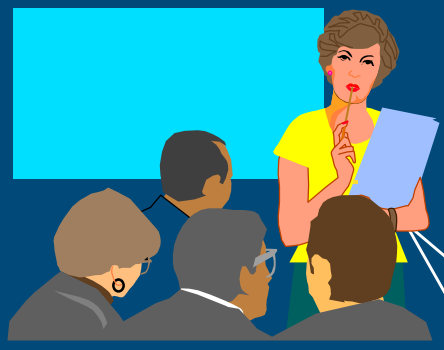


IQUIP / SCK
QWE 2000 - 10



Ah, you mean a knife!

Group sessions (identify - value - specify)



ADVANTAGES

- Direct communication;
- All parties participate;
- Understanding needs various stakeholders:
  - business
  - user
  - administrator
- Participants agree upon needs.

IT facilitates the sessions,
IT does not participate !!!

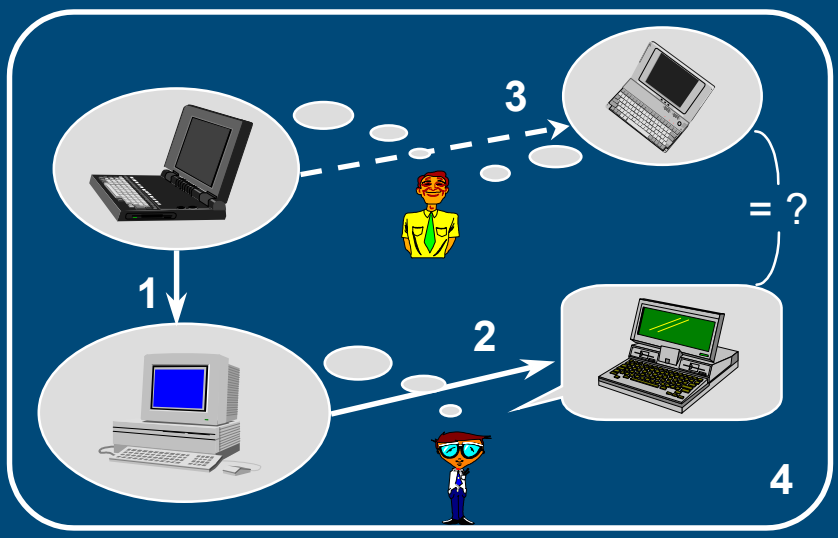
Functionality (Tailor-Made)

Money Constraint	Time Constraint
• Budget: 1.500.000 Euro • Experience base: - 1.500 Euro / fp - IA => 80% functionality - 40 fp / entity • Scope: ☞ 20 entities ((1.500.000 / 1.500) * 0,8) / 40	• Budget: 6 months / 10 people • Experience base: - 2 days / fp - IA => 80% functionality - 40 fp / entity • Scope: ☞ 12 entities (((6 * 20) * 10) / 2) * 0,8) / 40

Functionality (determine - eliminate)

RISKS	MEASURES
• Consistency in deliverables in development	☑ System development method ☑ Standard for deliverables ☑ Inspections (Fagan, …)
• Quality of the data	☑ Continuos data check ☑ Remove redundancy ☑ Control update

Project challenge



Which properties?

Extended ISO-model

Reliability
- Maturity
- Fault tolerance
- Recoverability
- Availability
- Degradability

Efficiency
- Time behaviour
- Resource behaviour

Maintainability
- Analysability
- Changeability
- Stability
- Testability
- Manageability
- Reuseability

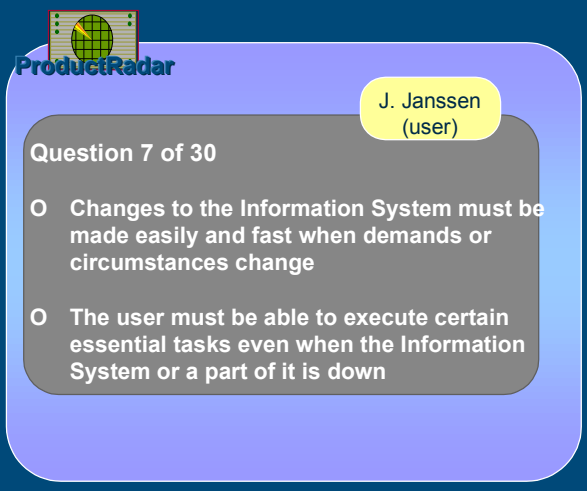
Functionality
- Suitability
- Accuracy
- Interoperability
- Compliance
- Security
- Traceability

Useability
- Understandability
- Learnability
- Operability
- Explicitness
- Customisability
- Attractivity
- Clarity
- Helpfulness
- Userfriendlyness

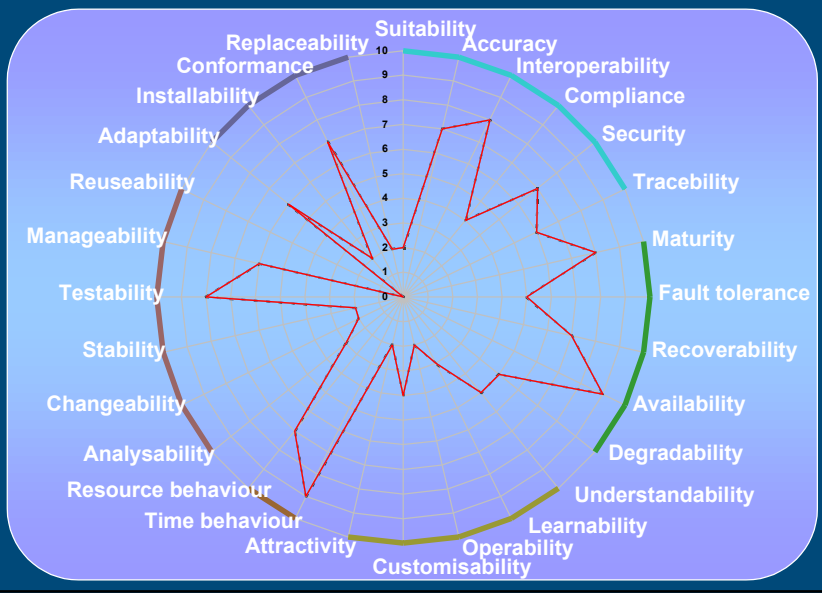
Portability
- Adaptability
- Installability
- Conformance
- Replaceability

Quint II

Make choices



Quality Radar



Product Risk Matrix

Product Risk Matrix	Functionality				Quality Aspects							
	consistent definitions	quality of data	inefficiency in the project	exceeding of the budget	standard screen lay-outs	context related help sceens	at least 50 users in one session	authorisation of transactions > 50k	screen changes < 0.5 seconds	couple to financial admin.	data definitions as in business model
Organisation Model								✓			
AO procedures								✓			
System Architecture							✓			✓	
Overview functions	✓										
Overview data	✓	✓								✓	✓
Function / Data Matrix	✓										
Man Machine Interface					✓	✓					
.....											

Project strategy

DEFINE MEASURES

- Adjust project approach
- Preventive / detective
- Project leader makes an 'all-in package'

- Quality plan:
  - standards
  - check & test activities
  - techniques
  - settlement findings

TUNE MEASURES

- Discuss risks
- Available budget
- Wanted "coverage"
- Determine impact of changes

RADAR

ADVANTAGES

- Product quality discussible and measurable
- Improved communication
- Structured project approach
- Risks ⇨ 'deliverables'
- Clear choices
- "Maintainable" strategy

RISKS

- Practise mechanically
- "Overkill" supposed risks
- Difficult project start up
- Each project starts again

Process (Risks)

IDENTIFY
VALUE
SPECIFY

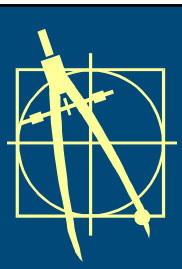
- Scope;
- Development platform;
- Project organisation;
- Schedule;
- Resources;
- Quality plan;
- Security plan;
- ...

DETERMINE
ELIMINATE

- Risk analysis;
- Budget (time / money);
- Measures;
- Metrics;
- ...

PROJECTPLAN

Metrics



Functional Size

- Function Points (IFPUG, NESMA, …)
- # Components
- # User Tasks

Productivity

- Hours / Activities
- Changes
- Management overhead

Quality

- Defects
- Inspection, review

Productivity Attributes

- Development Platform
- Tools, Methods, Standards, ...
- Resources, Skills
- Project constraints

More Metrics (Tailor-Made): GQM method


IQUIP
Customer Satisfaction
Business Need
Understanding
Requirements
transformed to
Design
Group sessions
Product Risk Matrix
business criteria for success
technical criteria for success
Customer satisfaction
contributes to
influencing the process
depends on
depends on
Quality Radar
Metrics
Quality aspects
supports
Development process
Expectations
Project Management
IQUIP / SCK
QWE 2000 - 25


IQUIP
… the 5 steps …
IQUIP / SCK
QWE 2000 - 26

Quality Tailor-Made

… don't schoot
at everything that moves …

- From expectations by way of risks to (additional) measures
- Measures directly related to 'project deliverables'
- Wittingly adjust 'deliverables' / project approach
- Supports communication (with all participants)
- Learning effect

QA

Thank you for your attention

QUESTIONS !
&
ANSWERS ?

Ton Dekkers
a.j.e.dekkers@iquip.nl
qtm@interdependent.nl

QWE2000 Vendor Technical Presentation VT1

Mr. Robin Bortz (Radview)

"WebLoad Integrity Testing for e-applications"

Key Points

- Point 1...
- Point 2...
- Point 3...

Presentation Abstract

WebLoad is the only testing tool that unifies load, performance and functional testing into a single process for shortened development cycles and unmatched verification of scalability and integrity prior to deployment. WebLoad verifies Web application scalability by generating a load composed of Virtual Clients that simulate real-world traffic. Users create JavaScript-based test scripts that define the behavior of the Virtual Clients. WebLoad executes these test scripts and monitors the Web application's performance providing real-time graphical and statistical results and comprehensive reports

1. Cruise Control: A goal seeking performance-testing scenario The WebLoad Cruise Control module enables you to test your Web server by specifying the performance goals that you want to achieve, and viewing the way your system meets those goals.

2. Functionality under load: Integrity testing for e-applications WebLoad delivers full Document Object Model (DOM) access. By creating and analyzing the DOM for every Virtual Client during a test Session, WebLoad is able to verify each success and failure and present detailed information to you about each transaction.

3. XML Testing: B2B verification WebLoad provides full support for work with the XML Document Object Model. Using XML DOM objects, WebLoad Agendas are able to both access XML site information, and generate new XML data to send back to the server for processing.

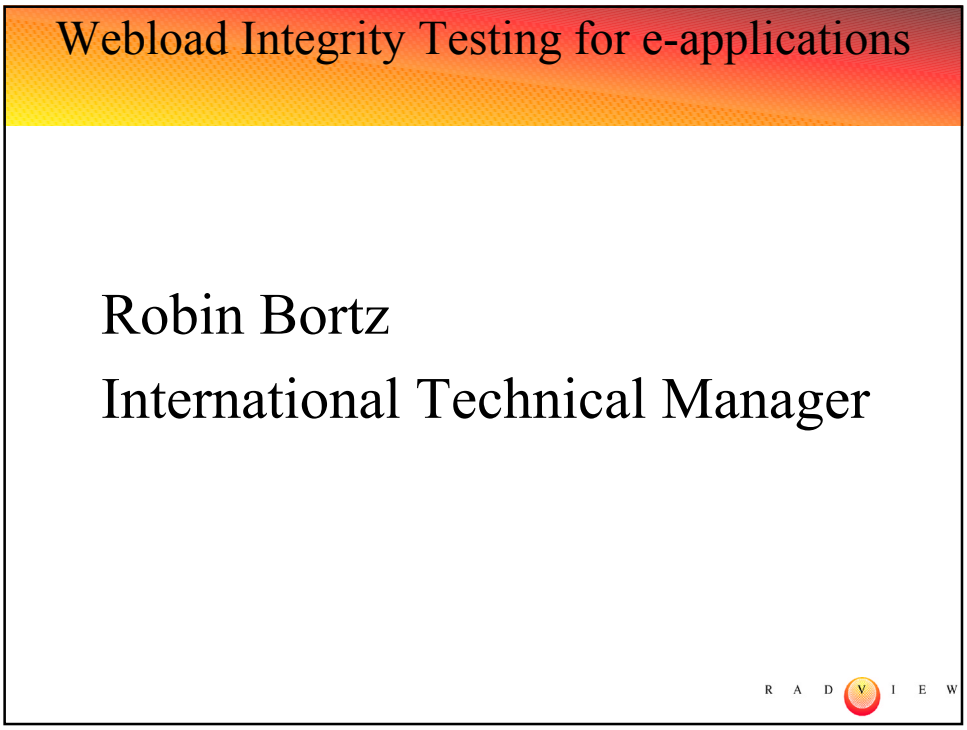
4. Beyond HTTP and SSL: FTP and SMTP WebLoad JavaScript provides direct object access to any component that has a COM wrapping and an Idispatch interface. WebLoad supports full Java access from your JavaScript Agendas. Full Java support means that your WebLoad Agendas not only test access time to an

HTML page, but also invoke and run Java classes used by the Java applications embedded within an HTML page.

About the Speaker

Speaker Bio


WebLOAD 4.0
R A D V I E W
R A D V I E W


Webload Integrity Testing for e-applications
Robin Bortz
International Technical Manager
R A D V I E W

Agenda

- Radview
- Testing
- Webload
- Demo's
- Resource Manager
- Summary

R A D V I E W

RadView: facts……

- Founded by Internet and Testing Developers in 1996
- WebLoad launched March, 1997.
- Headquarters : Boston, MA USA
- Main shareholders: Computer Associates, Formula group, RAD group
- IPO Completed 8th August 2000, raised $40m
- UK & German offices now running

R A D V I E W

RadView News

- Selected to test Windows 2000
- Crossroads 2000 A-List Award
- Java Developers Editor's Award
- Selected by Microsoft BackOffice to test Windows DNA performance
- Selected to test NetAid

Industry Accolades

"The RadView tools have incredibly easy-to-use interfaces, and I found them to be very easy to insert during the early stages of the development cycle."

"RadView's WebLoad product line enables organizations to meet e-Business quality challenges, not only by offering highly effective scalability testing solutions, but also by extending the reach of such solutions throughout the development lifecycle and the virtual enterprise."

"With WebLoad Resource Manager 4.0, RadView offers support for the next generation of Web application development technologies and processes, building on the company's vision of offering solutions that can be shared by all actively involved in the development lifecycle."

Bad news travels...

More Web sites turn to test tools

Free Britannica.com site shuts out some first-day surfers

By Stacy Collett

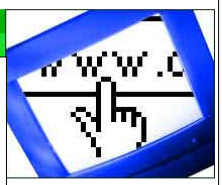
Hours after Encyclopaedia B
new, free Web content yeste

Some visitors to Britannica.c
reach the site at all, while oth
found it wouldn't accept sear
many users were turned awa

Schwab's Web Site Troubled Again

By Reuters

NEW YORK (Reuters) – Charles Schwab customers for the third day in a row had trouble Friday morning funneling stock trades through the Web site of the No. 1 U.S. discount and Internet brokerage, according to messages posted on the Internet by Schwab customers.



R A D V I E W

Web Application are:

- **Complex and distributed involving many components and connecting interfaces**
- **Composed of "emerging" technologies**
- **Developed under enormous competitive pressures**
- **Exposed to unpredictable loads and stresses**
- **Continuously updated and evolved**
- **Expected to be Business-critical in reliability**

R A D V I E W

Webload is…

The premier tool for performance, scalability and functional testing under load of Web applications

R A D V I E W

What is WebLoad ?

- WebLoad is the premier Web application scalability testing solution
- WebLoad integrates load, performance, functional and stress testing
- WebLoad is designed specifically for testing Web applications
- WebLoad automates Web application testing
- WebLoad is a distributed application

R A D V I E W

WebLOAD 4.0

- Offers an easy to use, wizard driven method to load testing: short learning curve.
- Offers the long term power: through JavaScript.
- Offers the tester the ability to develop an enhanced skill set in true internet language: JavaScript.
- Enhances co-operation between development team components.

R A D V I E W

Webload: Powerful and Cost effective Internet Testing Solutions

1. Cruise Control
   - A goal seeking performance-testing scenario
2. Functionality under load
   - Integrity testing for e-applications
3. XML Testing
   - B2B verification
4. Beyond HTTP and SSL
   - FTP and SMTP

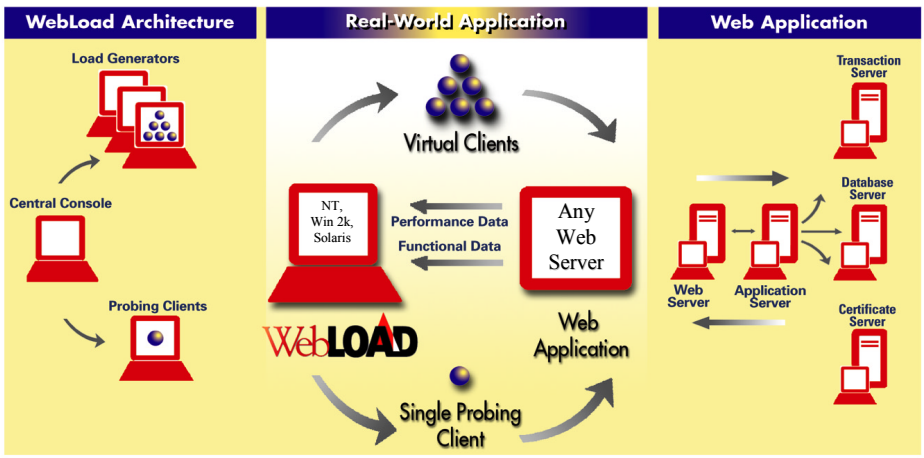
R A D V I E W

WebLoad 4.0

- Industry standards
  - JavaScript
- Performance
  - Memory Usage
- Innovation
- Ease-of-use
- Value

R A D V I E W

Webload Architecture



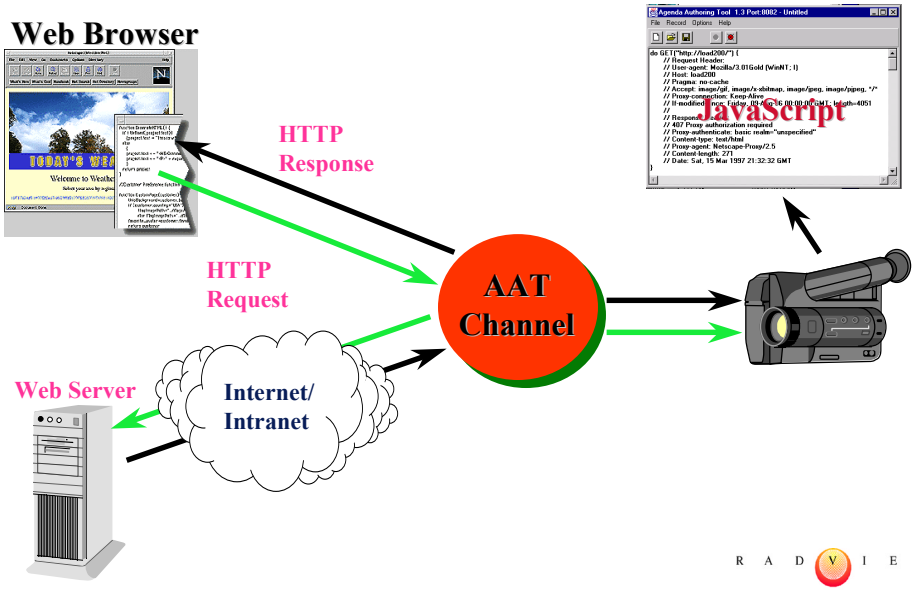
R A D V I E W

Full HTTP Support:

- Full HTTP 1.0 and 1.1 Support
- Persistent Connections (keep-alive)
- Cookies
- Proxies
- Redirections
- SSL (All Major Cipher Suites)
- Client Certificates
- Dynamic HTML (DHTML) Support

R A D V I E W

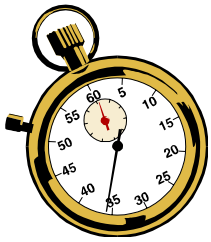
Agenda Authoring Tool



R A D V I E W

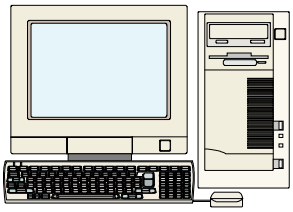
Performance Measurements

- Load generator & probing client : default 80 parameters
  - Load Size
  - Throughput
  - Rounds per second
  - Transactions per second
  - Connection trials
  - Succeeded connections
  - Failed connections
  - Response time
- User-Defined: Isolate bottlenecks
- Server under test measurements
  - Memory usage
  - CPU usage



R A D V I E W

Platforms



- Console
  - Windows NT 4.0
  - Windows NT 5.0
  - Windows NT 3.51
  - Windows 95
  - Windows 98
  - Windows 2000
- Load Generators
  - Windows NT x.x, Windows 2000
  - Solaris, Q4 2000 - IBM RS6000, Linux

R A D V I E W

Demonstration

- Recording load and functional tests
- Executing load tests and reporting on results using cruise control
- WebLoad with data drilling functional testing under load
- Scheduling tests with the scheduler

R A D V I E W

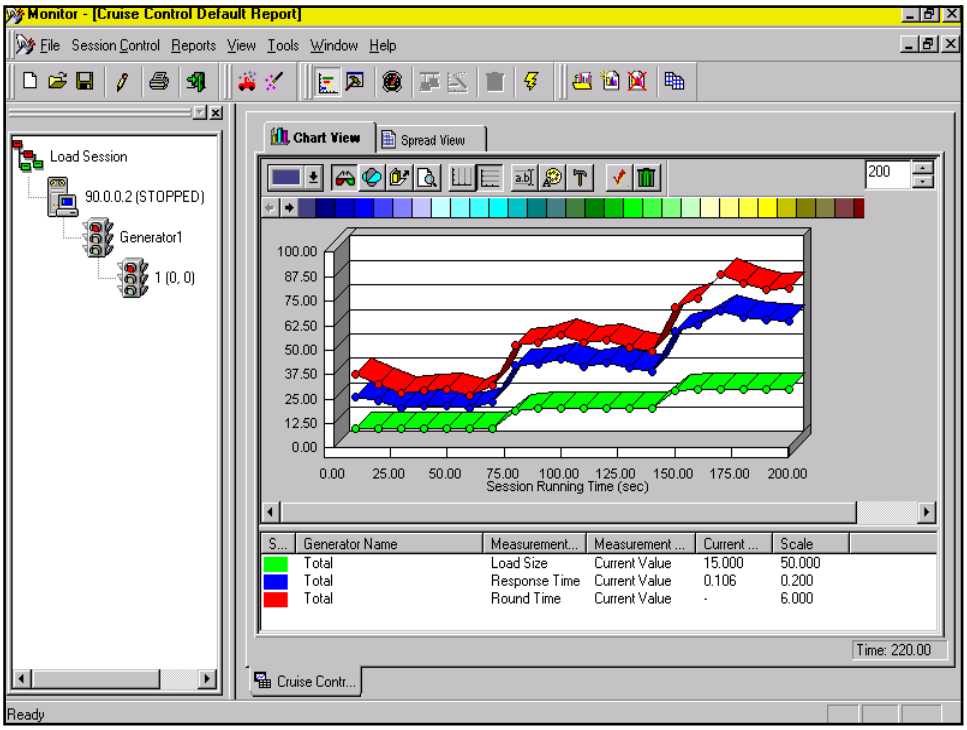
Cruise Control

- Automated performance threshold testing
- Tester defines performance goals
- Cruise Control increases Virtual Clients until performance criteria are met
- Intelligent Load Generation

R A D V I E W

Demo's

R A D V I E W



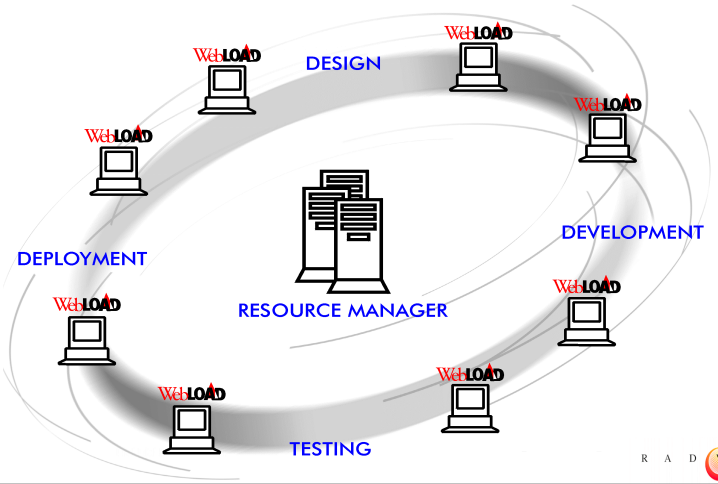
Highlights

- Cruise Control -Goal seeking performance testing:
- Throttle Control
- Simulate different user connection speeds & browsers
- Real Time Reports
- JavaScript as the scripting language.
- DOM support: functional testing under load
- Synchronization points: for distributed testing/stampede testing
- Automatic recognition and test creation Wizards for DataDriven, Session ID.

R A D V I E W

Lifecycle Testing Access

Solution for E-Business Success

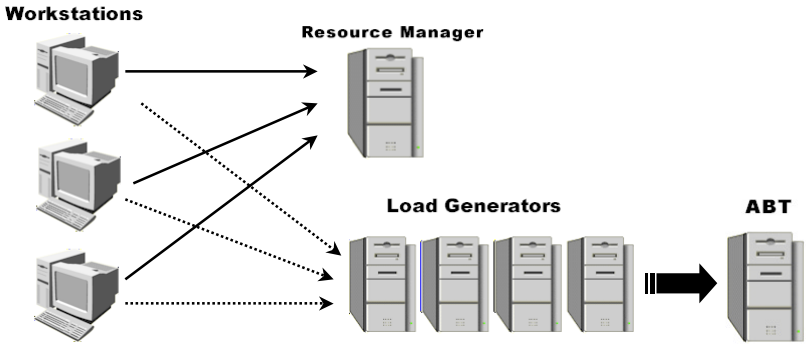


R A D V I E W

Webload Professional

R A D V I E W

Webload Resource manager



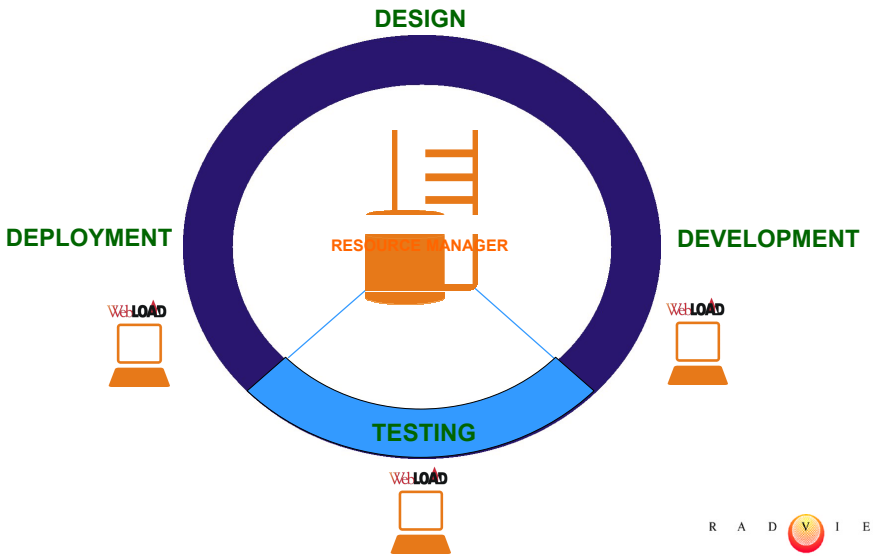
R A D V I E W

Resource Manager Features

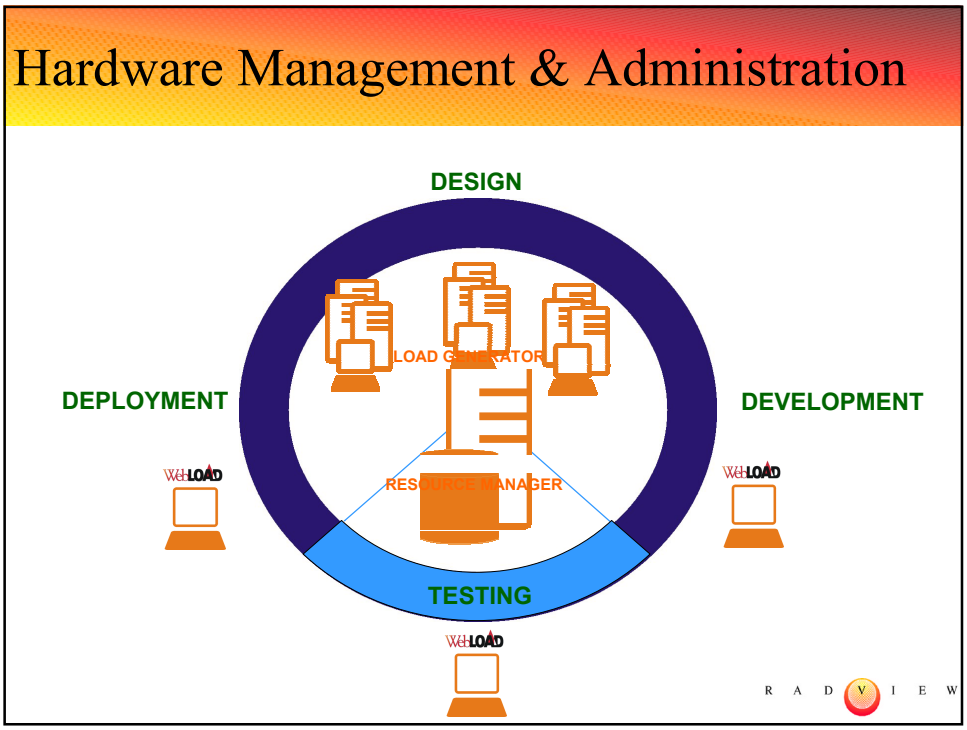
- Architectural evolution
- Dynamic license serving
- Hardware management
- Administration
- Allows sharing of LoadGenerator Hardware and WebLoad licenses across multiple sites/teams

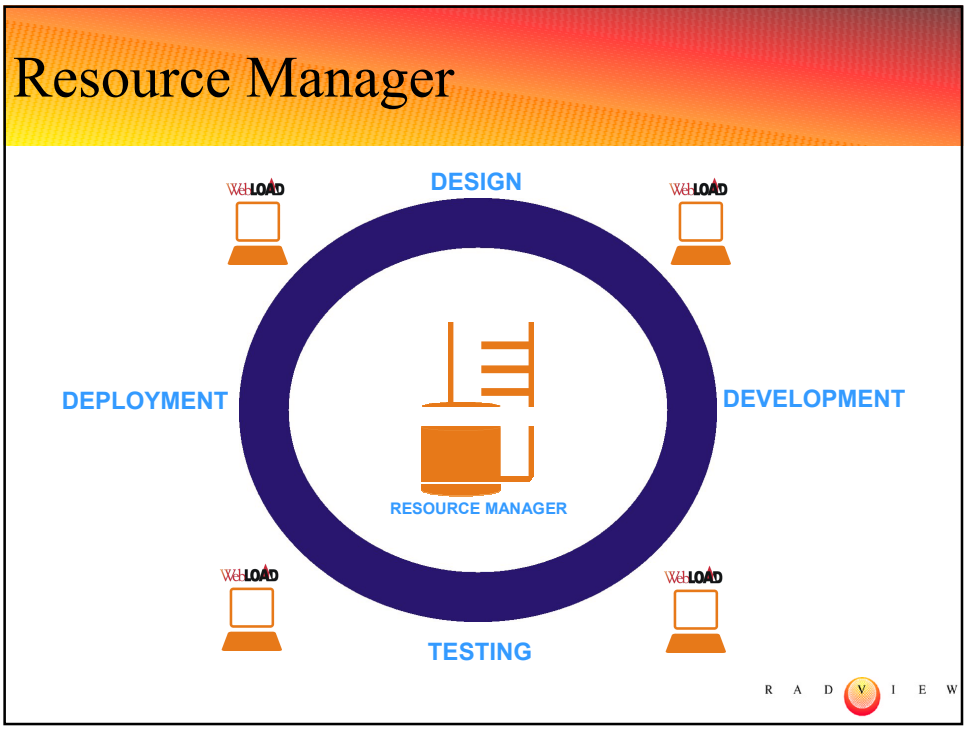
R A D V I E W

Dynamic License Serving



R A D V I E W


Hardware Management & Administration
DESIGN
DEPLOYMENT
DEVELOPMENT
TESTING
RESOURCE MANAGER
R A D V I E W


Resource Manager
DESIGN
DEPLOYMENT
DEVELOPMENT
TESTING
RESOURCE MANAGER
R A D V I E W

QWE2000 Session 2T

Mr. Tom Hazdra & Lubos Kral
[Czech Republic]
(CertiCon)

"Enhancing the Integration Testing of Component-Based Software"

Key Points

- Integration testing
- Distributed software
- Testing automation

Presentation Abstract

Integration testing is one of the basic stages in the waterfall product life cycle model. It is widely understood as testing of combined parts of an application to determine if they function together correctly. In this paper we identify what we see as a life cycle of software integration testing activities and discuss its enhancements from the perspective of testing software consisting of functionally distributed components. Our contribution is based mainly on testing the application projects from the area of cardiological therapy and diagnostics that we are developing for Vitatron Medical. It also explores knowledge gathered on projects for other companies including Medtronic and Rockwell Automation.

About the Speaker

Tom Hazdra received MSc. in Computer Science from CTU Prague in 1993. He works as research fellow at CTU since 1996. His area of research includes Distributed Artificial Intelligence, namely the coordination issues in multi-agent systems, and the software diagnostics. Since 1995 he participated in various software testing projects for Rockwell Automation, Vitatron and Medtronic.

Lubos Kral received MSc. in Control Engineering from CTU Prague in 1993. He received PhD. in Artificial Intelligence and Biocybernetics from CTU Prague in 1999. He works as research fellow at CTU since 1996. His area of research includes intelligent robotics, namely the mobile robot navigation. Since 1998 he participated in various software testing projects for Vitatron and Medtronic.

ENHANCING THE INTEGRATION TESTING OF COMPONENT-BASED SOFTWARE

Tom Hazdra and Lubos Kral

The Gerstner Lab, Czech Technical University
CertiCon Corp.

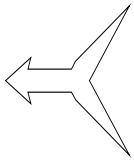
OUTLINE

- ❑ **INTEGRATION TESTING LIFECYCLE**
- ❑ **FORMAL METHODS**
- ❑ **COLLABORATION TEST AUTOMATION**
- ❑ **GUI AUTOMATION TOOLS**
- ❑ **CONCLUSION**

INTEGRATION TESTING

SOFTWARE DEVELOPMENT LIFECYCLE

- ❑ REQUIREMENTS
- ❑ SPECIFICATIONS
- ❑ DESIGN
- ❑ DEVELOPMENT
- ❑ UNIT TESTS
- ❑ INTEGRATION TESTS
- ❑ SYSTEM TESTS
- ❑ QUALIFICATION



INTEGRATION TESTING LIFECYCLE

- ❑ INSPECTIONS OF TESTING RESOURCES
- ❑ IDENTIFICATION OF REQUIREMENTS
- ❑ GENERATING TEST CASES
- ❑ TEST EXECUTION
- ❑ REGRESSION TESTING

PARTITIONING THE INTEGRATION TEST DESIGN

- ❑ **UI TESTS**
- ❑ **FUNCTIONAL TESTS**
- ❑ **COLLABORATION TESTS**

FORMAL METHODS

- OFFER MEANS FOR MODELING THE SOFTWARE DESIGN PRIOR TO IMPLEMENTATION
- ASSIST TO DETERMINE COMPLETENESS AND CONSISTENCE OF DESIGN SPECIFICATIONS
- PROTOTYPE VERIFICATION SYSTEM (PVS):
  - SOFTWARE SPECIFICATION REPRESENTED BY A THEORY, DEFINITIONS, LEMMAS AND THEOREMS IN FIRST ORDER LOGIC
  - INFERENCE ENGINE BASED ON LOGICAL RESOLUTION

DEADLOCK EXAMPLE

```
% definition of theory
Deadlock : THEORY
BEGIN

% definition of types
M : INTEGER
N : INTEGER
GenProcess : TYPE = {p : I NTEGER | p < N+1}
Process : TYPE = {p : GenProcess | p > 0}
GenSource : TYPE = {s : INTEGER | s < M+1}
Source : TYPE = {s : GenSource | s > 0}
SourceUsedBy : TYPE = [Source → GenProcess]
ProcessWaiting : TYPE = [Process → GenSource]
State : TYPE = [# UsedBy : SourceUsedBy, Waiting : ProcessWaiting #]

% definition of variable types
gp : VAR GenProcess
p,p1,p2: VAR Process
s,s1,s2 : VAR Source
st : VAR State
```

DEADLOCK EXAMPLE

% definition of axioms
initstate : STATE = (# UsedBy := λp:0, Waiting := λp:0)
UseSource (st,p,s) : STATE = IF ∀ s_1 : ((UsedBy(st)(s_1) = p) ⊃ s < s_1)
THEN IF ∃ p_1 : UsedBy(st)(s) = p_1
THEN st WITH [(Waiting)(p) := s]
ELSE st WITH [(UsedBy)(s) := p]
ENDIF
ELSE st
ENDIF
BadUseSource (st,p,s) : STATE = IF ∃ p_1 : UsedBy(st)(s) = p_1
THEN st WITH [(Waiting)(p) := s]
ELSE st WITH [(UsedBy)(s) := p]
ENDIF
DirectBlock (st,p,p_1) : BOOL = ∃ s : (s = Waiting(st)(p) ∧ UsedBy(st)(s) = p_1
Block (st,p,p_1) : INDUCTIVE BOOL = DirectBlock(st,p,p_1)∨ (∃ p_2 : DirectBlock(st,p,p_2) ∧ Block(st,p_2,p_1))
Deadlock(st) : BOOL = ∃ (p : Process) : Block(st,p,p)

DEADLOCK EXAMPLE

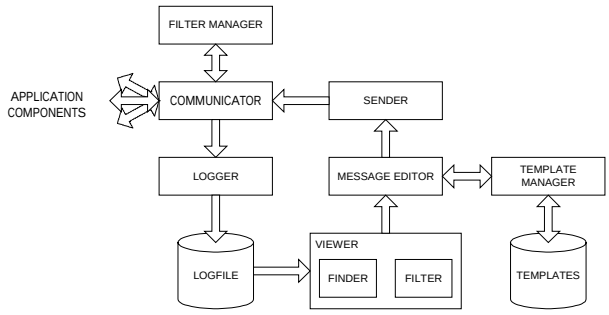
% definition of lemmas
Lemma1 : LEMMA ∀p,p_1 : ¬(DirectBlock(initstate,p,p_1))
Lemma2 : LEMMA ∀p,p_1 : ¬(Block(initstate,p,p_1))
Lemma3 : LEMMA ∀p_1,p_2,s_1,s_2 : p_1 ≠ p_2 ∧ s_1 ≠ s_2 ⊃

Deadlock(BadUseSource(BadUseSource(BadUseSource(initstate,p_1,s_1),p_2,s_2),p_2,s_1)p_1,s_2))
END Deadlock

COLLABORATION TESTS

- **MESSAGE SEQUENCE TESTS**
  - BASED ON UNIFIED MODELING LANGUAGE SEQUENCE DIAGRAMS
  - VERIFY HIGH-LEVEL COLLABORATION SCENARIOS
  - PASSIVE MESSAGE MONITOR USED FOR AUTOMATION
- **PRE- AND POST- CONDITION TESTS**
  - BASED ON SOFTWARE DESIGN SPECIFICATIONS
  - VERIFY LOW-LEVEL COLLABORATION SCENARIOS
  - ACTIVE MESSAGE MONITOR USED FOR AUTOMATION

ACTIVE MESSAGE MONITOR



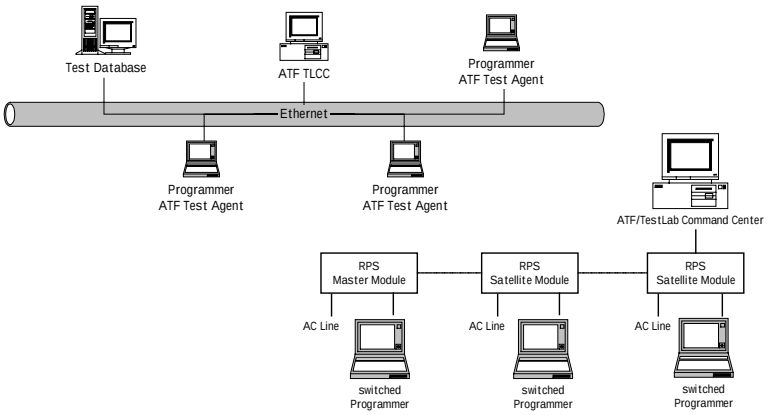
ACTIVE MESSAGE MONITOR

1) Application is brought to a certain state via standard interfaces available to the user.
2) Active message monitor is used to interrupt the work of particular components.
3) Active message monitor is used to evoke the desired pre- or post-condition state by simulating the interrupted components via absorbing and inserting the necessary messages from and to the system, respectively.
4) A message log is examined by the message log parser to verify the tested component responses.

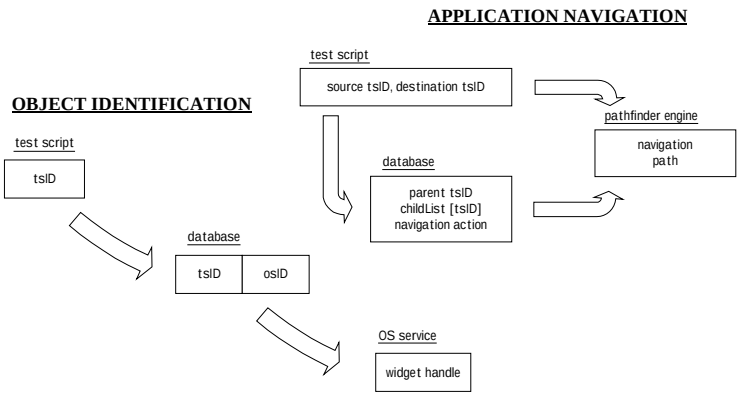
GUI TEST AUTOMATION TOOLS

- **TEAM TEST, WINRUNNER, ATF**
- **FUNCTIONAL TEST AUTOMATION**
- **SCRIPTING LANGUAGES,**
- **SENSITIVE TO GUI CHANGES**
- **SYNCHRONIZATION PROBLEMS**

GUI TEST AUTOMATION ENVIRONMENT EXAMPLE



SYMBOLIC APPROACH TO NAVIGATION AND IDENTIFICATION



CONCLUSION

- **INTEGRATION TESTING IN WIDER PERSPECTIVE**
- **FORMAL METHODS**
- **AUTOMATING THE GUI AND COLLABORATION TESTS**

CONTACT

CertiCon a.s.
Odboru 4/278
120 00 Prague 2
Czech Republic

phone: +420-2-24917207
fax: +420-2-24917209
email: hazdra@certicon.cz
http://www.certicon.cz

ENHANCING THE INTEGRATION TESTING OF COMPONENT-BASED SOFTWARE

TOM HAZDRA AND LUBOS KRAL

The Gerstner Lab, Czech Technical University, Technicka 2, CZ-166 27 Prague 6

CertiCon corporation, Odboru 4, CZ-120 00, Prague 2

e-mail: hazdra@certicon.cz
phone: +420-2-24917207-8
fax: +420-2-24917209

ABSTRACT

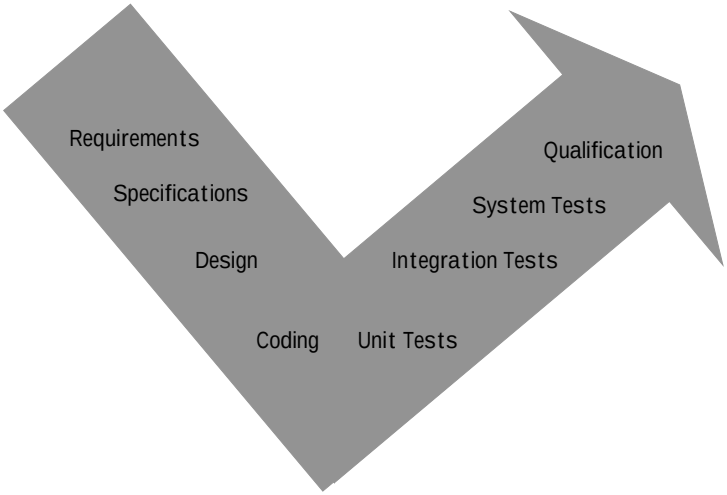
Integration testing is one of the basic stages in the waterfall product life cycle model. It is widely understood as testing of combined parts of an application to determine if they function together correctly. In this paper we identify what we see as a life cycle of software integration testing activities and discuss its enhancements from the perspective of testing functionally distributed, component-based software. Our contribution is mainly based on testing the application projects from the area of cardiac therapy and diagnostics that we are developing for Vitatron Medical but it also explores knowledge gathered on projects for other companies including e.g. Rockwell Automation.

INTRODUCTION

Software testing is not only the process of executing programs with the intention of finding errors. It is rather a set of activities related to each stage of a software product life cycle that are performed by software engineers in order to enhance the product quality. ***Integration testing*** is often defined as testing of combined parts of an application to determine if they function together correctly. The objective of integration testing is to find bugs related to interfaces between modules as they are integrated. According to our statement on software testing, we see integration testing in wider perspective, e.g. as the

integration testing itself as defined above plus a set of integration-related activities bound to the stages that precede the integration testing stage in a software product life cycle. In this way, we can define a set of areas in which we partition the software integration testing process:

- ***Inspections of testing resources***
- ***Identification of testing requirements***
- ***Generating test cases***
- ***Test execution***
- ***Regression testing***



Software Product Life Cycle (V – model)

INSPECTIONS OF TESTING RESOURCES

Inspections are an integral part of the review process when the test engineers review the product documentation (requirement, specification and design documents) with special effort on the software integration. The special intention of these reviews is to ensure the testability of the requirements with respect to the particular demands on the integration testing (testing automation, testing coverage, etc.). Typical results of these inspections are additional software requirements (e.g. requests for special drivers to access embedded features via testing automation tools).

IDENTIFICATION OF TESTING REQUIREMENTS

Testing requirements are identified from the testing resources, i.e. from the system and software requirements and from the software design documentation (UML diagrams, specification documents). Identification of testing requirements is a process of selecting the subset of existing system and software requirements to be verified in the integration test and defining an additional set of requirements based on the design

specifications. All the testing requirements are thus traceable either from the existing requirements or from the design specifications.

Generating Test Cases

Important part of integration testing is the test design, i.e. generating a set of test cases that sufficiently test the integration of application components. As the test designers should find a balance between the tested code coverage and acceptable cost of the integration testing, techniques such as boundary value testing are employed, and either the statement, branch or path coverage is to be reached.

Test Execution

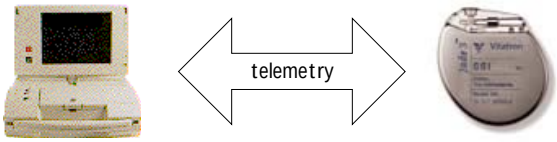
Test execution is the final test case execution. It is often provided in incremental way – the application is tested in small chunks so that errors are easy to observe, isolate and correct. The test cases that are often some collaboration scenarios are partitioned and tested partially. The partial tests are then gradually put together to complete the designed test cases. In the partial tests, the missing components are replaced with stubs that simulate the activity behind their interface. The incremental execution not only helps to simplify the testing but also enables to start the test execution sooner when some components are not yet developed. The incremental execution can lead to test automation since it is an often strategy to automate as much as possible of all repetitive actions.

Regression Testing

After fixing the detected errors by changing the code all the related tests should be regressed, i.e. re-executed. The usually time and manpower consuming test regression is a great opportunity for testing automation.

The Integration Testing Process

Our current work partially is aimed at integration testing of software for pacemaker programmers. The software enables the cardiologist to maintain the pacemaker in the patient's body by adjusting its internal parameters or therapies such as output voltage and pulse width or the pacing mode (e.g. single or dual chamber) and by analyzing the intra-cardial ECG.



Programmer and the pacemaker

The objective of this paper is to introduce the techniques we use to enhance the integration testing of the programmer software that is based on functionally distributed,

loosely coupled components that communicate by the means of message exchange. The software is additionally characterized by the facts that it is object-oriented and running on a special multi-processor platform, it is real-time, and that it must be highly reusable and, as a software for medical purposes, highly reliable. Our testing is provided in accordance with the ISO 9001, ISO 12207, and FDA quality regulation standards for software design and development.

Our integration testing process is organized in the following life cycle:

1. **Input resources review and analysis** – inspections of testing resources and identification of testing requirements.
2. **Test planning** – detailed planning of test activities including decisions about testing methodologies and test coverage and specification of the testing environment.
3. **Test design** – specifying the detailed structure of the integration test, generating the set of the application test cases.
4. **Test execution** – executing the test cases by the means of automated and manual testing, invoking changes to developed software and/or documentation.
5. **Regression testing** – re-testing of the functionality that may have been degraded by modifications made to the software to fix the found errors.
6. **Test evaluation** – summarizing, analyzing and reporting the results.

Further in this paper, we concentrate on the design phase of the integration testing life cycle and discuss the following issues that are helping us to make the integration testing process faster and/or more effective:

- Using formal methods for verification of component collaboration design
- Automating the verification of component collaboration sequence diagrams
- Using existing GUI testing automation tools to support the integration testing.

PARTITIONING THE INTEGRATION TEST DESIGN

According to our experience, we organize the integration test in three general test groups, the user interface (UI) tests, the functional tests, and the collaboration tests. The main reason for such organization is the test group independence with respect to the testing techniques and tools used.

UI tests verify the correct appearance of application screens and GUI objects and also the correctness of the GUI navigation. The tests are based on the UI specification documents. Our intention is to get most of the UI tests automated because of the relative stability of the GUI interface and since we always expect regression tests for various hardware platforms (with different displays). Another reason for UI test automation is that its results such as the test scripts used to access the GUI objects and to navigate through the application are reused for the functional tests.

Functional tests verify that the features of the integrated software behave as desired and are generated from the system and software requirements. They are gray-box tests since they mostly use special drivers or interfaces (prepared only for the testing purposes and not available in the product) to access the software internals to be verified. As the

regression of functional testing is of high probability especially because of the frequent software reuse, usually only small percentage of the tests is executed manually.

Collaboration tests verify that the interactions between the integrated software components are as designed in the software design specification (SDS) documentation. We employ the combination of a complete component set testing (to verify the high-level UML sequence diagrams) and a limited component set testing (to verify low-level UML sequence diagrams and component pre- and post-conditions). Whereas the UI and functional tests can be automated using commercially available GUI automation tools, to enable the collaboration test automation, special development efforts must be provided.

FORMAL METHODS FOR VERIFICATION OF COMPONENT COLLABORATION DESIGN

To start the collaboration tests at earlier development stages when the implementation has not yet begun we need a model of the designed software. As a modeling tool we have chosen the Prototype Verification System – PVS. It is a software package that employs formal methods to state problems and find their solutions. Generally, a problem in PVS is represented in the first order logic by a theory and a set of lemmas. A theory is a set of mathematical definitions and axioms that formally describe the problem's domain of discourse. Lemmas and theorems are formal descriptions of the assumptions that are to be proven to solve the problem. PVS, having a formal representation of a problem uses an algorithm based on logical resolution to find the proof of the lemmas and theorems within the specified theory.

The general obstacle of using system like PVS is the ability of translating the software specification in first order logic. An example of a problem representation is introduced below for the deadlock problem. In this representation, the axiom ***BadUseSource*** represents a wrong memory allocation that can be detected by proving the third lemma.

```
% definition of theory
Deadlock : THEORY
BEGIN

% definition of types
M : INTEGER
N : INTEGER
GenProcess : TYPE = {p : I NTEGER | p < N+1}
Process : TYPE = {p : GenProcess | p > 0}
GenSource : TYPE = {s : INTEGER | s < M+1}
Source : TYPE = {s : GenSource | s > 0}
SourceUsedBy : TYPE = [Source → GenProcess]
ProcessWaiting : TYPE = [Process → GenSource]
State : TYPE = [# UsedBy : SourceUsedBy, Waiting : ProcessWaiting #]

% definition of variable types
gp : VAR GenProcess
p,p1,p2: VAR Process
s,s1,s2 : VAR Source
st : VAR State
```

```
% definition of axioms
initstate : STATE = (# UsedBy := λp:0, Waiting := λp:0)
UseSource (st,p,s) : STATE = IF ∀ s1 : ((UsedBy(st)(s1) = p) ⊃ s < s1)
                             THEN IF ∃ p1 : UsedBy(st)(s) = p1
                                  THEN st WITH [(Waiting)(p) := s]
                                  ELSE st WITH [(UsedBy)(s) := p]
                                  ENDIF
                             ELSE st
                             ENDIF
BadUseSource (st,p,s) : STATE = IF ∃ p1 : UsedBy(st)(s) = p1
                                THEN st WITH [(Waiting)(p) := s]
                                ELSE st WITH [(UsedBy)(s) := p]
                                ENDIF
DirectBlock (st,p,p1) : BOOL = ∃ s : (s = Waiting(st)(p) ∧ UsedBy(st)(s) = p1
Block (st,p,p1) : INDUCTIVE BOOL = DirectBlock(st,p,p1)∨ (∃ p2 : DirectBlock(st,p,p2) ∧ Block(st,p2,p1))
Deadlock(st) : BOOL = ∃  (p : Process) : Block(st,p,p)

% definition of lemmas
Lemma1 : LEMMA ∀p,p1 : ¬(DirectBlock(initstate,p,p1))
Lemma2 : LEMMA ∀p,p1 : ¬(Block(initstate,p,p1))
Lemma3 : LEMMA ∀p1,p2,s1,s2 : p1 ≠ p2 ∧ s1 ≠ s2 ⊃
   Deadlock(BadUseSource(BadUseSource(BadUseSource(initstate,p1,s1),p2,s2),p2,s1)p1,s2))
END Deadlock
```

PVS formal representation of a deadlock problem

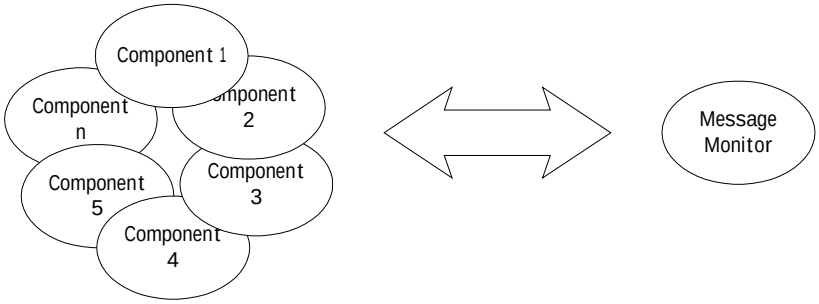
Once we have the formal representation of the specification we can attempt to prove that it is correct. The PVS algorithm is, however, rather used in a reverse way – to verify the completeness of the specification. If we have only a draft specification of the theory that contains only the core definitions and axioms, and a set of theorems that should be valid then the PVS leads us to find the minimal theory that satisfies the goal theorem.

AUTOMATING THE COLLABORATION TESTS

Collaboration tests verify the component interactions by observing the information exchange directly on the component interfaces. The main goal is to cover all possible scenarios that can occur in the application under test. We attempt to reach this goal in two ways – by examining the message sequences under the desired scenarios and by examining the specified component pre- and post-conditions.

The message sequence tests are provided to verify the rather high-level UML sequence diagrams. A desired scenario is invoked and the inter-component information exchange is observed to verify the correct behavior. The tests are usually provided with a complete set of components. We use a ***passive message monitor*** device included in the application to log the messages being exchanged together with their time stamps. The passive monitor can insert verification tags in the log for easier locating the particular message sequence. The produced message log is then examined whether it conforms to the specified message sequence. A Perl-based message log parser is used to verify the sequence either by comparing it to the pre-verified gold file or by verifying the message

internal data. As mentioned before these tests are used only for the high-level scenarios, mainly because of an unfeasible numbers of scenario variances at deepest levels.
The component pre-conditions (i.e. specific component reactions on certain states of the application) and post-conditions (states of the application after the specific actions provided by particular components) are examined in two ways – either manually or automatically. Manual tests are done by compiling pieces of application brought to a desired state by hard coding the initializing message sequence. In this way, we can use only limited set of components that are involved in the particular test case and, therefore, these tests can be done at early development stages when not all components are



1) Application is brought to a certain state via standard interfaces available to the user.
2) Active message monitor is used to interrupt the work of particular components.
3) Active message monitor is used to evoke the desired pre- or post-condition state by simulating the interrupted components via absorbing and inserting the necessary messages from and to the system, respectively.
4) A message log is examined by the message log parser to verify the tested component responses.

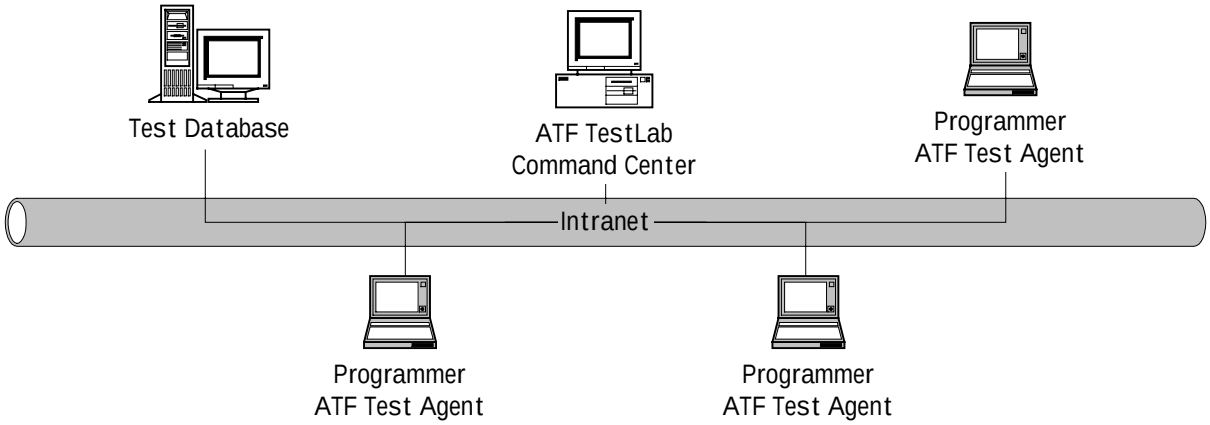
General pre- and post-condition automated test design

available. Automatic tests are supported by the ***active message monitor*** device that allows us to bring the application in a desired state by simulating responses of selected components. It is an extended passive message monitor that can switch the application components on and off and that can receive, debug and send specific messages. The general use of an active message monitor is sketched in the above figure.

USING GUI TESTING AUTOMATION TOOLS

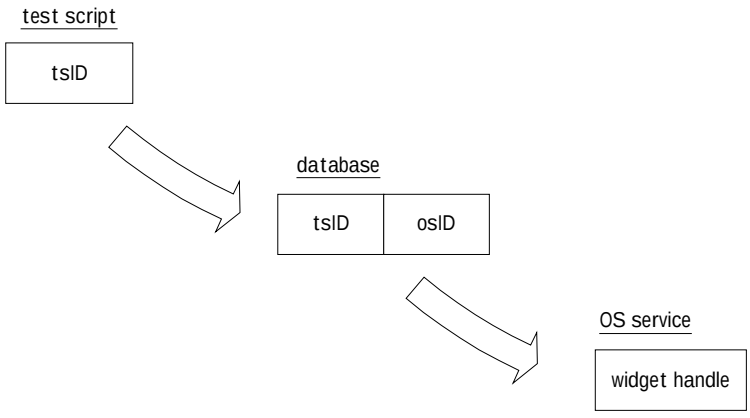
Incremental development where new features are gradually added between the individual increments requires large amounts of integration tests to be regressed. As our human resources for testing are limited we tend to automate as much tests as possible. GUI testing automation tools such as ATF (Softbridge), WinRunner (Mercury Interactive), or SQA Team Test (Rational) are used to create programs (called test scripts) written in a tool-specific programming language, which simulate the user behavior by generating desired GUI events and verify the software application responses. The software functionality is verified by detecting the GUI events created as a response to the simulated events and comparing them to the expected values. Verification means of the

tools include support actions for checking window/object existence, verifying the attributes of specific objects (size, position, label, text, etc.) or matching bitmaps.
Currently we use the ATF GUI testing automation tool of Softbridge mainly for the functional black-box tests where software features are accessed either by standard GUI or by special GUI drivers created only for the purpose of testing. These drivers are created to access either the features that do not have their GUI (e.g. audible feedback, telemetry, or internal state variables), or the features that GUI has not been yet developed in the particular increment. Although it is possible to control the message monitor by ATF we do not use it for the component collaboration tests because of the real time synchronization problems.



GUI testing automation environment setup

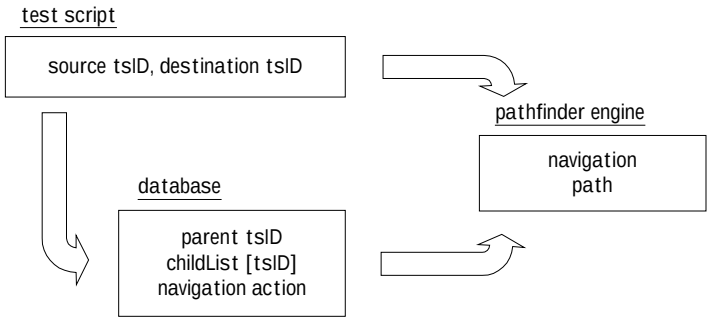
To reduce the total testing time, we run the tests on several programmers in parallel. The test scripts that are located on the database server are executed on the programmers connected via intranet to the ATF TestLab Command Center, which controlls the test script distribution to the programmers available and manages the test reports.



GUI object identification

While using the GUI test automation tools we experienced several problems. First of all, these tools are quite sensitive to the changes of the GUI design during software development such as additions of new GUI objects and windows, changes of specific

GUI controls, etc. In order to avoid this sensitivity we adopted a symbolic approach to GUI object identification and to GUI navigation. Within the test scripts we use symbolic identifiers, which we map to the operating system identifiers that are hard coded by agreement with the developers. The operating system services can map these identifiers to the display manager widget handles, which are then used by the automation tool to access the particular GUI objects. Similarly, the navigation hierarchy and the navigation actions are represented in the database and the navigation path is detected by the



Detecting the navigation path

pathfinder engine test script. First, it detects the positions of the source and destination objects in the navigation hierarchy, then it finds the shortest navigation path, and finally it returns the navigation path in the form of sequence of GUI events to be generated.

Another problem is that of accessing the custom controls, i.e. the objects derived from the standard controls available in the presentation manager of the particular operating system. To be able to access the custom controls by the particular GUI test automation tool, special interfaces must be developed. This is an extra effort that must be taken in account when planning the development activities. Developing the custom controls may result in the problem of consuming the presentation manager system messages, which is a programming technique rarely used by developers to avoid undesired displaying problems. The automation tools, however, need the information contained in these messages to control the application GUI. Consuming the system messages can therefore lead to disabling the tool functionality.

TESTING EVALUATION

Generally, we evaluate the testing process by analyzing the defects and the testing costs. We base the evaluation on the data collected in the test summary reports delivered to our customers.

The defect analysis gives us the areas that need improvement. We categorize the defects by feature (documentation, GUI, component functionality, integration of components) and by severity. We follow the collected defect numbers in each testing phase and reflect the findings in the phases that follow. One of the most informative metrics on the testing process effectiveness is the ratio between the defects found by executing the designed tests and by providing „random“ tests (performed also by the developers).

The testing costs analysis helps us to plan the future testing process resources. We express the costs in man/days spent on the testing activities (test design and preparation, test execution, test regression, test design rework) again categorized per feature.

CONCLUSION

In this paper, we presented our approach to enhance the integration testing of component-based software. We identified the phases of the integration testing process and focused on three main areas, the exploration of formal methods for verification of component collaboration design, the automation of component collaboration verification, and the use of GUI automation tools to support the integration tests.

REFERENCES

Beizer B.: **Black-Box Testing.** Wiley, 1995.

Horch J.W.: **Practical Guide to Software Quality Management.** Artech House, 1996.

Kit, E.: **Software Testing in the Real World.** Addison-Wesley, 1995.

Myers, G.J.: **The Art of Software Testing.** Wiley, 1976.

Owre S., Rushby J., Shankar N., and Calvert D.: **PVS: An Experience Report.** Springer Verlag Lecture Notes in Computer Science Vol.1641, pp.338-345, 1998.

Owre S., Shankar N., and Rushby J.: **PVS: A Prototype Verification System.** Proc. of CADE 11, Saratoga Springs, 1992.

Rakitin, S.R.: **Software Verification and Validation.** Artech House, 1997.

QWE2000 Session 2A

Mr. Olaf Mueller & Mr. Axel Podschwadek
[Germany]
(Siemens)

"A Step-to-Step Guide to Incremental Testing: Managing Feature Interaction for Communication Devices"

Key Points

- The focus is laid on restructuring the test process: Define clear test exit criteria. Divide integration into different levels. Define integration levels feature-oriented, not archtitecture-centric. Augment project management by an integration matrix. Move system test to the integration phase of each increment. Automate system test to enable regression. Change configuration management to distinguish between error handling and development of next increment. Change review techniques to deal with component additions. Change requirements engineering and change requests.

Presentation Abstract

In recent years pressure on the communication industry concerning a shorter time-to-market has been steadily growing. This demands for a significant reduction of project-specific development times. At the same time complexity is growing tremendously, especially for the software components. The interaction of the continuously growing number of new features reaches a dimension which is hardly manageable. As one solution to both problems incremental development processes have been proposed. They allow to serve the market at hardly any time by in the same time splitting up the system into pieces of manageable size. Furthermore, such processes permit a validation of the system from the user's perspective already in early stages of development. However, the effort and complexity of introducing an incremental process should not be underestimated. In this paper we propose a detailed roadmap for such an improvement goal by presenting well-defined and fine-granular steps of process change. They start from later stages of development and move then on to earlier stages. The focus is on test process improvements, but hints for all other relevant and concerned process areas are given as well. The work is based on experiences made with the development of communication devices within the Siemens company.

About the Speaker

Speaker Bio

SIEMENS

A step-to-step Guide to incremental testing managing feature interaction for Communication devices

Author	Axel Podschwadek		
Company	Siemens	Phone:	+49 / 2871 / 91-2438
Org.	ICM MD QM DP PSI	Fax:	+49 / 2871 / 91-2495
E-mail	axel.podschwadek@bch.siemens.de		

SIEMENS

Introduction

Goals:

- shorter time-to-market
- high quality software, (Reason: no possibility for an update)
- reduce development effort

Methods:

- introduce an incremental development and test process
- splitting up system into smaller pieces
- dividing development process into differant cycles
- validate system from user´s perspective
- automate testing

SIEMENS

Content

Steps to incremental testing

- Step 1: Define clear test exit criteria
- Step 2: Divide integration into different levels
- Step 3: Define integration levels feature-oriented, not architecture-centric
- Step 4: Improve project management by an integration matrix
- Step 5: Move system test to the integration phase of each increment
- Step 6: Automate system test to enable regression

- Conclusion

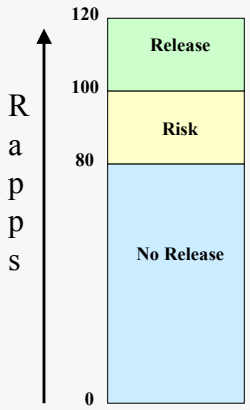
SIEMENS

Step 1: Define clear test exit criteria

- Measuring every phase with metrics
- Rate every metric to a global scale

- Ranges of the Rapps-scale:
  - 0..79: No release to next phase
  - 80..99: Risk-area
  - 100..120: Release to next phase

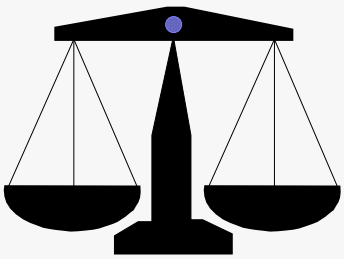
- Advantages:
  - Easy to communicate
  - Established in short time



SIEMENS

Step 1: Define clear test exit criteria

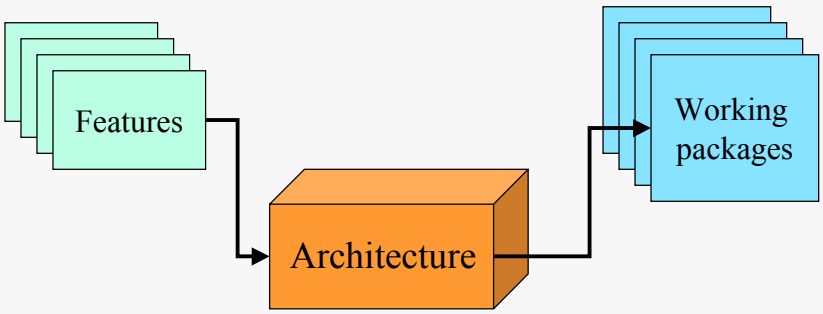
- Metrics are measured to:
  - tracking a project and
  - approve the product
- Metrics could not be only formal. They need to have a technical background.
- Examples for metrics for testing:
  - Test coverage
  - predicting bugs
  - bug fixing percentage
  - and a lot others



SIEMENS

Step 2: Divide integration into different levels

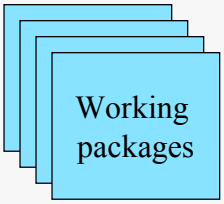
- Specify all requirements as a list of features
- Deriving the architecture
- Generating a list of working packages



SIEMENS

Step 2: Divide integration into different levels

- Organize all working packages in steps
  - specify each package
  - estimate effort of each package
- Generate an integration-plan:



Int 1	OS, Cores
Int 2	First call
Int 3	Application
Int n	Feature oriented packages

SIEMENS

Step 3: Define integration levels feature-oriented, not architecture-centric

- basis of the integration matrix:
  - Input data:
    - A list of all features
    - Planned integration step for each feature (see step 2)
    - A list of all working packages
    - Estimated effort for each working package
  - Output data:
    - effort per feature
    - effort per integration step

SIEMENS

Step 3: Define integration levels feature-oriented, not architecture-centric

Example:

	Working package A	Working package B
Feature 1	X	
Feature 2		X

Meaning:

Feature 1 is associated to working package A,
Feature 2 is associated to working package B

SIEMENS

Step 3: Define integration levels feature-oriented, not architecture-centric

			A	B	C	D	E	F	G		
Features	**Ready at step (down)**	**Effort (right)**	60	20	20	30	10	15	40	**3.**	**4.**
Feature 1	1		X		X	X		X		4	35
Feature 2	3		X				X			2	15
Feature 3	2		X		X			X		3	20
Feature 4	3		X		X		X		X	4	40
Feature 5	1		X		X	X		X	X	5	55
Feature 6	2		X	X						2	30
Number associations	**1.**		6	1	4	2	2	3	2		
Effort per association	**2.**		10	20	5	15	5	5	20		

SIEMENS

Step 3: Define integration levels feature-oriented, not architecture-centric

Using the full example (previous page):

Step	Action	Example
1.	Counting all associations per column.	The computed numbers are right from 1.
2.	Computing the associated amount. For example through dividing the estimated effort by the number of associations (linear approach)	The computed numbers are right from 2.
3.	Relate for every association the associated amount	The computed numbers are below 3.
4.	Add the associated amount per feature.	The computed numbers are below 4.

SIEMENS

Step 3: Define integration levels feature-oriented, not architecture-centric

- Generated results:
  - After performing all steps the „effort per feature" is calculated
  - This can be summarized to effort per integration step by adding each effort of a single feature

Integration step	Features	Computed effort
1	Features 1 and 5	90
2	Features 3 and 6	50
3	Features 2 and 4	55

SIEMENS

Step 4: Improve project management by an integration matrix

- Generating a project plan out of effort:
  - having constant time:

Resources		
	Resources	Resources
90h	50h	55h

  - having constant resources:

time

Resources		
90h	50h	55h

SIEMENS

Step 5: Move system test to the integration phase of each increment

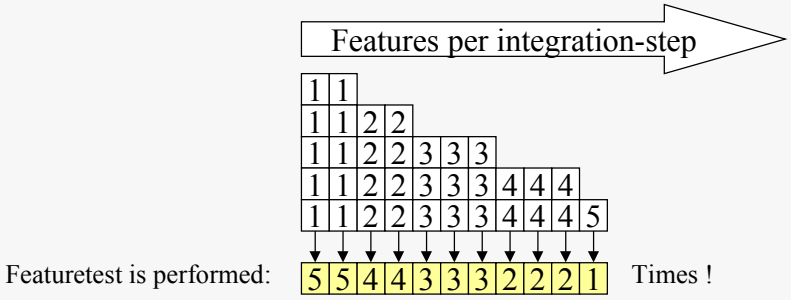
- Performing tests on a product thats implementation is not complete is difficult:
  - There are bugs discovered in areas, that are still not complete
  - testers need a test-specification in which is discribed what is testable
  - Only effort lies in early discovered bugs
- All necessary information can be derived out of the integration matrix

Features	Ready at step
Feature 1	1
Feature 5	1
Feature 3	2
Feature 6	2
Feature 2	3
Feature 4	3

SIEMENS

Step 6: Automate system test to enable regression

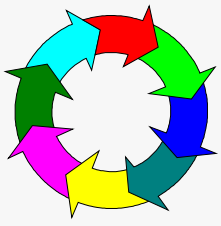
Performing regressiontests for each integration-step means, to repeat the same test for several times.



SIEMENS

Step 6: Automate system test to enable regression

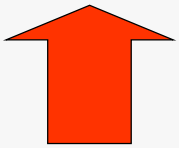
- The next step is to establish a recorder tool.
  - Performing the tests once,
  - storing all relevant data leads up to
  - a complete reusable set of tests



SIEMENS

Conclusion

- Benefits from incrementell testing and using the integration matrix:
  - planning easier, on a more detailed level
  - mostly all relevant information can be calculated with a tool
  - very efficient guideline for blackbox-testing
  - additionally stability in project management data
    - team size can be set out of information
    - structure can be chosen more easily
- Increasing project-process performance by a factor 4.



A Step-to-step Guide to incremental testing managing feature interaction for Communication devices

Axel Podschwadek, Siemens AG

Abstract

In recent years pressure on the communication industry concerning a shorter time-to-market has been steadily growing. This demands for a significant reduction of project-specific development times. At the same time complexity is growing tremendously, especially for the software components. The interaction of the continuously growing number of new features reaches a dimension which is hardly manageable. As one solution to both problems incremental development processes have been proposed. They allow to serve the market at hardly any time by in the same time splitting up the system into pieces of manageable size. Furthermore, such processes permit a validation of the system from the user's perspective already in early stages of development. However, the effort and complexity of introducing an incremental process should not be underestimated. In this paper it is proposed a detailed roadmap for such an improvement goal by presenting well-defined and fine-granular steps of process change. They start from later stages of development and move then on to earlier stages. The focus is on test process improvements, but hints for all other relevant and concerned process areas are given as well. The work is based on experiences made with the development of communication devices within the Siemens company.

Introduction

In recent years pressure on the communication industry concerning a shorter time-to-market has been steadily growing. Siemens develops and manufactures in Bocholt digital cordless phones for the mass market. The code is stored in ROM on board of the processor. So there is no possibility to make an update of the software having millions of phones at the homes of the users.

The mass market forces development departments to bring up new products in very short cycles with an increasing number of features. Out of this reason complexity is growing and managing the project became almost impossible. Making routhly estimations based on our experience is possible, but not sufficient enough. For a more detailed prediction we need for our long running phase a better statement. This demands for a significant reduction of project-specific development times. At the same time complexity is growing tremendously, especially for the software components. The interaction of the continuously growing number of new features reaches a dimension which is hardly manageable.

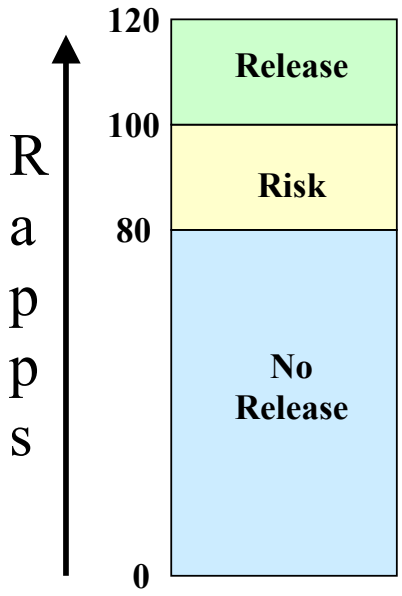
As one solution to this problem incremental development processes have been proposed. They allow to serve the market by a shorter period of time by splitting up the system into pieces of manageable size. This is reached by dividing the development process (organized as a usual waterfall model) into several cycles, each cycle covering all development activities and resulting in a further increment of the system.

In addition, incremental processes permit a validation of the system from the user´s perspective already in early stages of development. However, the effort and complexity of introducing an incremental process should not be underestimated.

The process starts from later stages of development and move then on to earlier stages. Our experience shows that a well-structured and planed roadmap is absolutely essential for the success of such an improvement, as process changes have to be implemented and accepted by the entire development team.

Step 1: Define clear test exit criteria

In order to establish clear and separate test phases, each phase has to be terminated using definite and meaningful metrics. If this is neglected there is a great danger that all test cases are moved backwards to late system tests. Before being able to establish different test cycles for different increments it is essential to be used to such metrics as otherwise there is no culture to deal with early tests. Although it is well known than early error detection is very cost-effective, often tests are performed as late as possible.



To measure our products we use a scale to rate every metric. This scale is the Rapps-scale. Rapps means “Rathmer points on the Podschwadek scale” and describes a modell out of which everyone can derive, whether a software is ready to release or not.

The first range “No release” starts from 0 up to 80 Rapps. It is just the information, that only a minimum percentage of the metric is fulfilled. Above 80 Rapps up to 99 Rapps is the risk-area. Giving a product to a next phase would be risky. Sometimes it is necessary to take a risk, but normally taking a risk could be more expensive than developing a product to its end in the range of 100 up to a maximum of 120 Rapps.

Using this scale simplifies communication to the major project managers, whose background in software engineering disciplines is limited.

Beyond the Rapps-scale is a set of other metrics, that consists out of three categories:

- Test coverage
- Predicting bugs
- Removing known bugs

Test coverage is directly measured as an unweight criteria and additionally as an weight criteria named test intensity.

Predicting bugs is easy, if you use an easy model and much more complex if you try to be to precise. Out of the relation between number of found errors compared to testing effort there can be made a curve. This curve is predictable, and the more bugs you found, the better is this model working.

A second model gives out of predicted lines of code (LOC) and average bugs per LOC´s a second prediction about the complete number of bugs.

Out of the curve of known bugs combined with a time index you get a curve, that in the beginning is constantly growing, and after a maximum is nearly linear decreasing with an average flatness. Out of this model it also is easy to build a prediction model with ranges. After reaching the maximum a date for achieving a level of bugs, that could be tolerated can be calculated automatically.

All these methods are basis for every phase, because they are unified. Reaching enough Rapps for a phase is criteria before beginning another. This only means fulfilling the own work to completeness is mandatory. Doing the work only the half could never be tolerated.

Step 2: Divide integration into different levels

In the last years, it was possible to establish a new method to specify a detailed plan for integration SW-components on a low-level managing the great mass of data that is generated through this method.

It will be described how this method has been introduced in a heterogeneous environment of software development projects, what are the constraints for this method, how the method is performed and how the method fits with our culture of software development and quality management.

After a period of sporadic use, the method was recognised by the project managers as a powerful tool for making development effort more convincing to the planned periods of time for integration. In earlier days these integration steps have been planned as a steady period of time. Today this space is filled with the information that is derived out of the matrix.

Every project manager is trained and coached for his individual project and mostly all proects made use of the method. Using the integration matrix has been a standard for our development process.

The core activities are focused in our software quality and process improvement department, which has established, trained and coached all users.

Every project gets is requirements as a list of features that are planned to be implemented. This list of features will be specified in more detailed requirement-specifications. Derived out of this list an architecture will be designed (however cordless phones do not differ one from another so much) and a list of working packages is generated from the architect.

The list of working packages is analysed be the programmers and a meeting was held, where everyone gives a number for the planned effort. These numbers will be assessed and brought to one number for every single working package.

Putting all these numbers together gives an easy overview over the schedule of the project and considering the available resources calculates a straight forward milestone for finishing the project.

One problem that results out of this estimation process is the fact that you get numbers for working packages, but you don't get numbers for features.

For performing black-box-integration-tests it is necessary to have an idea, when each single feature is complete and testable. This information depends on the completeness of the working packages, but is not directly combined. The reason is a working package oriented mode of operation of the developers, that is derived out of the architecture.

Putting all these information together, is the major task of the integration-matrix.

Step 3: Define integration levels feature-oriented, not architecture-centric

The integration-matrix was invented while discussing about modultesting. Dividing each step of development, particularly the coding phase in shorter parts, leads in the direction of defining software-modules or software-units. Putting all units together leads to a complete whole.

A work group was build in order to discuss these methods above the range of one single project. After long meetings, suggesting various possible ways, the prototype of the matrix was generated on a Microsoft Excel-base. Managing a little number of working packages and features, this first approach was the beginning of a more comprehensive tool.

Although it was a comprehensive tool a lot of training and coaching was necessary.

An additional problem was the range of information that is stored inside this matrix. A standard project has no less than 150 features, combined with no less than 50 working packages makes a minimum of 7500 possible combinations that have to be analysed.

A Support for generating and handling each matrix for every project was arranged to ensure the proper use of the matrix, to accumulate the experience while using the matrix, to collect metric data and to help the project managers to make use the matrix.

Inputs of the matrix are:

- A list of all features
- Planned integration-step for each feature
- A list of all working packages (coding)
- Assumed effort for each working package

These information was associated in a grid through a label. The meaning of this label is, that the specific feature is associated to a working package.

	Working package A	**Working package B**

Feature 1	X	
Feature 2		X

This table shows an example of associating feature 1 to working package A, and associating feature 2 with working package B.

			A	B	C	D	E	F	G		
Features	**Ready at step (down)**	**Effort (right)**	60	20	20	30	10	15	40	**3.**	**4.**
Feature 1	1		X		X	X		X		4	40
Feature 2	3		X				X			2	15
Feature 3	2		X		X			X		3	20
Feature 4	3		X		X		X		X	4	40
Feature 5	1		X		X	X		X	X	5	55
Feature 6	2		X	X						2	30
Number associations	**1.**		6	1	4	2	2	3	2		
Effort per association	**2.**		10	20	5	15	5	5	20		

This next table shows a complete integration-matrix example. Any required input is given (features 1 to 6, date of ready at (integration-)step, working packages A to G, estimated effort for each working package).

Additionally all associations are entered.

Computing the matrix was performed in 4 steps (lighted in green big numbers):

Step	Action	Example
1.	Counting all associations per column.	The computed numbers are right from 1.
2.	Computing the associated amount. For example through dividing the estimated effort by the number of associations (linear approach)	The computed numbers are right from 2.
3.	Relate for every association the associated amount	The computed numbers are below 3.
4.	Add the associated amount per feature.	The computed numbers are below 4.

The result after this step is the effort per feature. The next step is to add up the amount per integration step.

This is done separately from the matrix.

Integration step	Features	Computed effort
1	Features 1 and 5	95
2	Features 3 and 6	50
3	Features 2 and 4	55

The result of the example is that the effort of integration-step 1 is nearly double the effort of step two or three. From this information there can be derived, that

1. having constant time, it is necessary to have additional resources or
2. having constant resources, it is necessary to have more time

for coding this first step of integration.

Step 4: Improve project management by an integration matrix

The feature-oriented view of integration levels and the architect view of the project plan are orthogonal to each other. Therefore they interact to each other and are often not without contradiction. Therefore it is essential to develop both concepts in parallel, and to detect and eliminate such inconsistencies.

As mentioned above, Siemens Bocholt has projects in increasing levels of complexity. So there is not one simple method planning and tracking the project. The project manager had to work out several information specific to the different kinds of sources, development environments and engineering methods.

When the integration-matrix is calculated, effort to each integration step is given, the next step is to start with the project management plan. The project manager had to transform the effort information using his resources to a time information. Without the matrix he would have to guess this effort.

Step 5: Move system test to the integration phase of each increment

Now – and only now - it is possible to introduce several separate development cycles by dividing the test process into different pieces.

To improve the process of testing software it is the best way to start with testing as early as possible. But each tester needs a guideline what he has to test, and it could be a more great effort to write this down, then to perform the test.

Additionally it is very frustrating for a tester to start testing on features that are simply not ready to test.

This information can also be derived out of the matrix!

Making a sort over the integration steps, all features (even then there is a great number) are in order of there completion. Knowing the dates of the integration steps, the tester has got a date based self increasing list of features, he had to test.

The data of the compare between planned and achieved functionality is input information of the status report for tracking the project. It has got the look of a milestone trend analysis and is generated automatically through this information.

Step 6: Automate system test to enable regression

As increments are defined via features and do not follow directly the component architecture, in each increment a new delta of code is added to each component. This has to be re-tested, as it is almost impossible to only retest the newly added additions. Therefore an efficient automation of regression testing is essential and easy to implement at this point.

To adopt this constraint to the process, it is important to realise, that most of the tests are performed on a user interface level. And even the tests that are not performed on this level could easily adopted to an interface, so they could be handled as being performed on an user interface level.

The next step is to establish a recorder tool. Performing the tests once, and storing all relevant data leeds you up to a complete re-usable set of tests.

This set of tests can be re-used for any iteration of regression testing.

Conclusion

The most benefits from the integration matrix are for derived out of benefits of the process.

- It is possible to plan easier, but on a more detailed level
- Mostly any relevant information can be calculated with a tool
- You got nearly for no additional effort a very efficient guideline for blackbox-testing
- Additionally stability of the project requirements
- team experience in
  - software engineering is culturiced
  - development environment is held stable
  - tools and methods are standardized
- team productivity factors
  - team size can be set out of information
  - structure can be chose more easily

The usage of this kind of a process fullfills our requirements in bringing up products more quickly than in the past. Even managing projects of the size we have today would not be possible. With constant resources we can handle up to four parallel developed projects, whereas in the past it was just one.

Managing feature-interaction has been a challenge of the last two years. Now we are up to optimize our tooling, that we will get more resoluted data of our results and process metrics. But derived out of our project scope we see an increasing of project-process-performance of about a factor 4.

Improving our test-process and doing some fine-tuning will be the issue of the following up activities.

QWE2000 Session 2I

Ms. Nicole Levy [France]
(Laboratore PRISM)

"Quality Characteristics to Select an Architecture for Real-Time Internet Applications"

Key Points

- software architecture
- quality attributes
- formal definition of architectural styles

Presentation Abstract

There is a general agreement on the fact that, at an architectural design stage, the consideration of quality issues are crucial for achieving the overall system quality goals.

The objective of this work is to present and discuss the experience obtained in applying the ABAS (Attribute-Based Architectural Style) technique, for the selection of the architecture of a stock exchange monitoring system.

Non-functional requirements for this application are high availability, platforms heterogeneity, distribution of clients, strict response time. The ABAS technique has been extended introducing the ISO 9126 quality model, to refine the initial high-level quality characteristic, considered relevant to the style, into measurable attributes. In this work we have formalized the definition of architectural styles with quality attributes. For the architecture of the data component, we considered the architectural style for indirect communication and component decoupling, favoring the availability attribute. Several patterns have been studied and evaluated with respect to the availability attribute: Publisher/Subscriber, Repository, Mediator and Broker.

The lessons learned were that the ABAS technique extended with the ISO 9126 quality model and the formal definition of the style, was useful to guide the selection of the architectural pattern. It helped to organize the specialist knowledge and to reason about the relevant quality characteristic of the pattern.

About the Speaker

Nicole Levy is professor at the University of Versailles, Saint Quentin en Yvelines. She leads a group on the design of software architecture based on the formalization of the development.

She collaborates since several years with F. Losavio, Professor at the Central University of Venezuela.

Previously she has been assistant professor in Nancy, France, working in the LORIA research center. She participated to the development of Proplane, a formal specification construction model independent of the language of expression, guiding the specifier in its work and memorising his/her expertise.

QUALITY CHARACTERISTICS TO SELECT AN ARCHITECTURE FOR REAL-TIME INTERNET APPLICATIONS

Losavio F., Matteo A., Ordaz Jr. O.
Centro ISYS, Universidad Central de Venezuela
Lévy N., Marcano-Kamenoff R.
Laboratoire PRISM, Université de Versailles

Architectural design

- Non-functional characteristics must be taken into account:

  performance, **security**, **availability, ...**
- Design strategies can be based on **architectural styles** or **patterns**
- The architecture is designed by stepwise transformations

Objective:

- Select the right architecture
- Take into account quality issues

$\Rightarrow$ apply the ABAS technique

ABAS= Attribute-Based Architectural Style

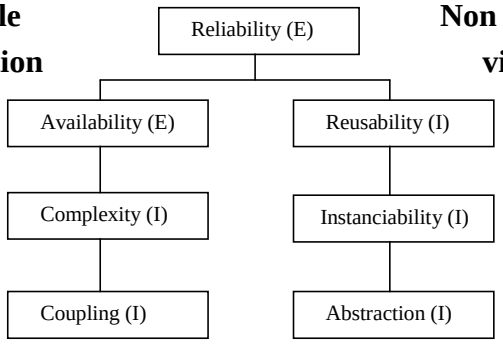
$\Rightarrow$ formalize quality attributes using the B language

➘ Case Study: Cyber Stock Exchange(CSE)

ISO 9126 quality attributes

Observable via execution

Non Observable via execution



Quality measures:

Ex : Availability $\frac{MTTF}{MTTF + MTTR}$

Architectural Styles

Description of

- the component types
- the connectors
- the topology
- global constraints

ABAS
Attribute-Based Architectural Style [Klein,Kazman 99]

- Problem description
- Quality attribute measures
- Architectural style
- Quality attribute parameters
- Analysis

The ABAS Structure

- 1. Problem description:

  Informal description of the design and analysis problem

  → the quality attribute of interest, ← *only one*

  → the context of use,

  → attribute-specific requirements.

- 2. Stimulus/Response attribute measures:
  - the stimuli to which the ABAS is to respond
  - the quality attribute measures of the response.

- 3. Architectural style:
  - its components,
  - its connectors,
  - their topology and
  - constraints on the style.
- 4. Analysis:
  - how the quality attribute models are formally related to the architectural style,
  - conclusions about “architectural behaviour”,
  - tradeoffs between the quality characteristics required and the measured properties affecting them.

Publisher-Subscriber ABAS

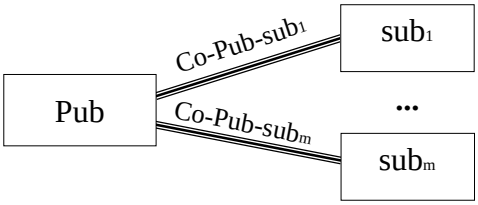
- Problem description:

  Allow multiple components to share data and be automatically updated
- Stimulus/Response attributes measures:
  - time of propagation of a change to the components
  - fraction of time that the system is working
  - number of components added, modified, deleted
  - complexity of these additions/modifications/deletions

- Architectural style
  - Components:
    - a publisher
      - maintains a registry of currently-subscribed components
      - sends changes notifications to the subscribers

        push model ↔ pull model
    - many subscribers
      - subscribe/unsubscribe to the publisher
  - Connectors:

    subscribers communicate with the publisher
  - Topology: *STAR*



- Constraints:
  - an object can be subscriber to many publishers
  - the only role of the publisher is to take the subscriptions and to send the change notifications

- Analysis
  - Data persistency: Transitory
  - Communication Protocol: Selective broadcasting
  - Dependencies from publisher Very high
  - Redundancy in data flow High

Selection of a style

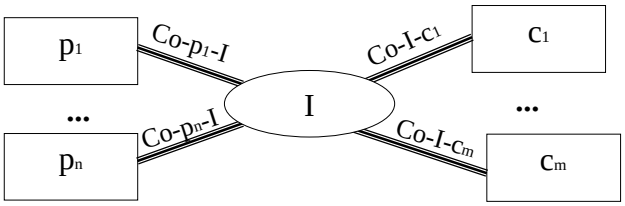
Styles studied and evaluated with respect to the **availability attribute**:

- Data indirection
- Publisher/Subscriber
- Mediator

Data indirection style

availability attribute for a consumer c_j communicating with the producers:

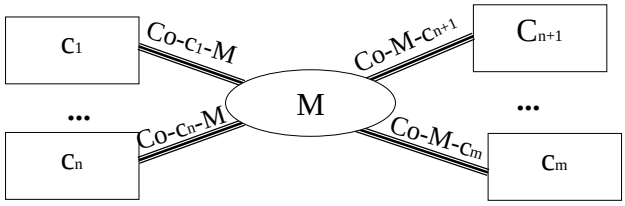
$$avail_{cj} = \Sigma \quad (avail(p_i) * avail(co\text{-}p_i\text{-}I)) \quad p_i \in producers * avail(I) * avail(co\text{-}I\text{-}c_j) * avail(c_j)$$



Mediator style

availability attribute for a colleague c_i communicating with a colleague c_j:

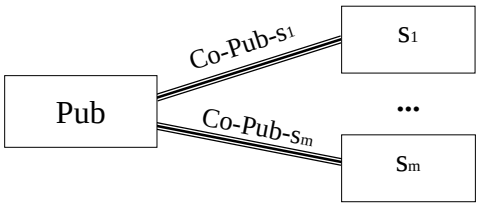
$$avail_{ci} = avail(c_j) * avail(co\text{-}c_j\text{-}M) * avail(M) * avail(co\text{-}M\text{-}c_i) * avail(c_j)$$



Publisher / Subcriber style

availability attribute for a subscriber s_i communicating with the publisher:

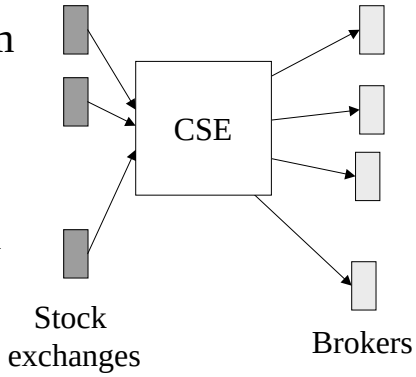
$$avail_{si} = avail(\text{ Pub }) * \ avail(\text{co-Pub-}s_i\ ) * avail(\ s_i)$$



Case study: CSE
Stock exchange monitoring system

Real-time data provider, to monitor stock exchanges for brokers and independent investors

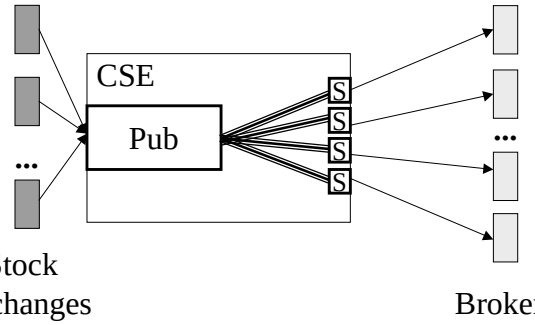
- a soft real-time problem
- non-functional requirements:
  - high availability
  - platforms heterogeneity
  - distribution of clients
  - strict response time



Application of Publisher/Subscriber style

availability attribute for a broker :

$avail_i = avail(Pub) * avail(co\text{-}Pub\text{-}s_i) * avail(s_i)$

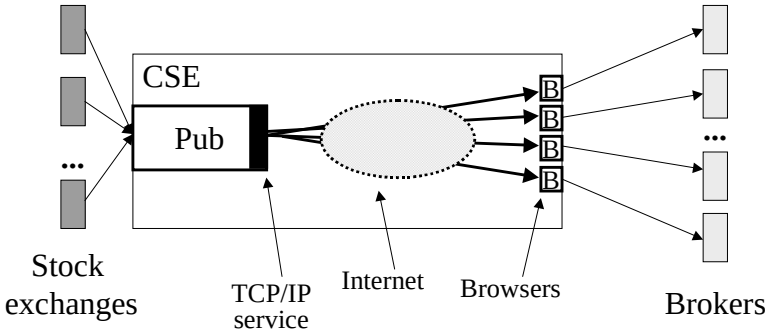


Reuse of Internet:

avail(co-Pub-s_i) = 1

availability attribute for a broker :

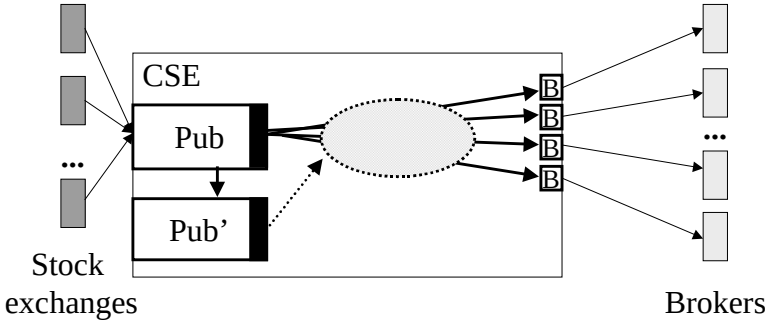
$avail_i = avail(Pub) * avail(B_i)$



Introduction of a redundant Publisher $\Rightarrow 2*$avail(Pub)

availability attribute for a broker :

$avail_i = 2*avail(Pub) * avail(B_i)$



Conclusion

- Architecture development by stepwise transformation
- Internet is reused to define the connector because of its availability
- Introduce a redundant publisher is a Middleware architectural pattern

QUALITY CHARACTERISTICS TO SELECT AN ARCHITECTURE FOR REAL-TIME INTERNET APPLICATIONS[1]

Losavio F., Matteo A., Ordaz Jr. O.
Centro ISYS, Facultad de Ciencias,
Universidad Central de Venezuela
Apartado 47567, Los Chaguaramos 1041-A, Caracas, Venezuela
{flosavio, amatteo, oordaz}@isys.ciens.ucv.ve

Lévy N., Marcano-Kamenoff R.
Laboratoire PRISM
Université de Versailles St.-Quentin,
78035 Versailles Cedex, France
{Nicole.Levy,Rafael.Marcano}@prism.uvsq.fr

Abstract

Performance, security and availability are important non-functional characteristics that must be present in real-time systems. The selection of a convenient architecture is an important step in achieving these quality goals. The use of an appropriate architectural style can simplify architectural design and subsequent software implementation stage. The overall quality goals are influenced by the structural characteristics or topology of the style. However, the problem on the selection of the right architectural styles according to the desired quality attributes is an open issue. The existing approaches lack of a standard and formal notation. They are limited to an informal description and examples of the application of a style. Quality issues are not explicitly considered. The main goal of this work is to propose an approach for the selection of software architectures based on quality characteristics. We present a process integrating the ABAS technique with the ISO 9126 quality model, taking advantage of their complementary strengths. The B formal language is used to formally describe architectural styles and their quality attributes. We describe and discuss an experience obtained in applying this process for the selection of the architecture of a market stock exchange monitoring system. One of the transformations introduces Internet as a communication medium.
Key-words: software architecture, real-time system, quality attribute, ABAS, architectural style

1. INTRODUCTION

Real-time time systems interact directly with electrical and/or mechanical devices, handling external events usually captured by sensors from the environment. They must be prepared to deal with safety-critical situations, which must be handled with strict timing and ordering constraints. They may vary in time and scope, but performance, security and availability are important quality or non-functional characteristics that must be present in such systems, whose failure may involve high costs, such as loss of human life.

An important step towards achieving the quality goals required by a real-time system is the selection of a convenient architecture for the corresponding software system [BCK 98], [BK 99]. Architectural design identifies the key strategies for the large-scale organization of the system under development [Kru 00], [SR 98], [Dou 99]. These strategies include for example, the mapping of a software package to processors, bus and protocol selection, at a quite low level of abstraction. Quality requirements are generally

[1] This research is sponsored by the CEE INCO SQUAD project EP 962019 and the CDCH ARCAS project No. 03.13.4584.00 of the Universidad Central de Venezuela

dealt with by a rather informal process during architectural design. Conventional object-oriented design methods [Rum et al 96], [Jac et al 92], [Kru 00] tend more on achieving the required system functionality, paying limited attention to quality requirements. Implicitly, the use of the object-oriented modeling approach guarantees to some extent the construction of reusable and flexible systems. Hence maintainability and reusability requirements are incorporated to some extent. However, only these quality characteristics are implicitly considered [Bos 00]. It is also of general agreement that the improvement of one quality attribute may negatively influence another one, so there must be a negotiation or tradeoff before building the final system. Otherwise, the inclusion of different quality requirements once the system is built, will be extremely costly. There are very few approaches to explicitly handle the conflicts in quality requirements during the architectural design stage [Bøe et al 99], [KK 00], [Bos 00], [Kas et al 98]. Consequently, the lack of a supporting method or systematization drives to design software architectures in an *ad-hoc*, intuitive, experience based manner, with the consequent risk of unfulfilling some of the system properties.

Few traditional software development methods deal explicitly with quality architectural design. New methods are arising.

A method, proposed by [Bos 00], considers the design of software architectures taking account of the quality requirements from the early stages of development. The architectural design process, seen as an optimization problem, is viewed as a function taking as input the functional requirements specification and generating as output the architectural design. In the first step, a first version of the architecture is produced, not accounting of the quality requirements. Then, this design is evaluated with respect to the quality requirements. Each quality attribute is given an estimated value. These values are compared with the values of the quality requirement specification. If all the values are as good or better than required, the architectural design process is finished. Otherwise, a second step transforms the initial architecture, during which, quality value for some attribute improves. This design is again evaluated and the same process is repeated, if necessary, until all quality requirements are fulfilled or until the software engineer decides that there is no feasible solution. In this case the software architect needs to renegotiate the requirements with the customer. Each transformation (quality attribute-optimizing solution), generally improves one or some quality attributes, affecting others negatively.

Another method, ATAM (Attribute Tradeoff Analysis Method), is similar to the one formulated by [Bos 00]. It is proposed by [Kaz et al 98] as a technique for understanding the tradeoffs inherent in architecture evaluation. The method provides a way to evaluate software architecture's fitness with respect to multiple competing quality attributes. Since these attributes interact, the method helps to reason about architectural decisions that affect quality attribute interactions. The ATAM is a spiral model of design, postulating candidate architectures followed by analysis and risk mitigation, leading to refined architectures. The technique used for helping the reasoning is based on Attribute Based Architectural Style (ABAS). A quality model for a particular quality attribute is

established to help in the selection of a style. An ABAS considers only one attribute at a time. If several attributes must be considered, the ABAS technique is reapplied.

Both methods are quite similar. However, one of the major differences between these approaches is that [Bos 00] method includes concrete guidelines on how to transform or refine the architecture in order to meet the quality requirements. ATAM, does not provide guidelines for refinement, concentrating instead more on the identification of the tradeoff points, e.g. design decisions that will affect a number of quality attributes.

For the purpose of this work, we have benefited from both approaches. We have applied the ATAM's ABAS technique to identify the relevant quality attributes, in order to evaluate the fitness of the proposed architectural style. However, since ABAS considers only one attribute at a time, we have used an extended ABAS [CLP 00], defining a quality model involving all the interesting attributes, according to the ISO 9126 model. In this way we have a global and better picture of all the involved quality attributes. On the other hand, we have used a formal approach based on the B language [Abr 96], similar to the transformation approach followed by [Bos 00], to formally justify the selection of the style and related patterns.

In what follows we will consider an architectural style [GS 96] or architectural pattern [Bus et al 96] as a general description of the pattern of data and interaction among the components. An informal description of the benefits and drawbacks of using the style is also provided [Bus et al 96], [KK 99]. A component of the style may be a design pattern, in the sense of [Gam et al 95].

The main goal of this work is to present and discuss the experience obtained in applying the ABAS (Attribute-Based Architectural Style) technique [KK 99], for the selection of the architecture of a market stock exchange monitoring system. This application is considered a soft real-time problem, in the sense that some of the events may miss their deadline, without affecting the whole system's behavior. The transformation process that undergoes architectural design is formally described by means of the B language. One of the transformations introduces Internet as a communication medium.

The structure of this paper is as follows. The first section introduces real-time monitoring systems. First, the requirements for the stock exchanges monitoring system are described. Then, a quality model is introduced, based on ISO 9126 model. A categorization of architectural styles for real-time systems is subsequently presented. The second section describes the process of selection of the architecture based on quality attributes. The ABAS technique is introduced. The third section illustrates the use of the B language to formally specify architectures with quality attributes. The whole process of applying the presented technique to select the architecture of the stock exchanges system is detailed in section 4. The last section discusses the acquired skills and advantages of the presented approach.

2. REAL-TIME MONITORING SYSTEMS

2.1 Requirements for a real-time stock exchanges monitoring system

The primary goal of a real-time monitoring system is to capture, analyze and broadcast events (data) in real-time. We are interested in *soft* real-time systems, where some of the events may miss their deadline, without affecting the whole system's behavior. The needs of real-time distributed applications running in heterogeneous environments interconnected by wide-area networks, have driven the requirements for an application that will be called CSE (Cyber Stock Exchange). Non-functional requirements for CSE are high availability, platforms heterogeneity, distribution of clients, reliable information with strict deadlines. It is known that these characteristics are not independent, and there must be a tradeoff to determine priorities.

The CSE system, as a real-time *data provider*, will monitor small and medium size Latin American stock exchanges for brokers and independent investors. An antenna (*feed server*) external to the system, provides the data (*feed*) to the CSE data server. A feed contains the relevant information of a stock exchange transaction. The clients (*brokers*), distributed in different geographical locations, are subscribed with the data server. When a change on the feed to which a client is subscribed occurs, the feed is broadcasted to him by the data server, according to a strict time delay. Since one of the requirements for the CSE platform is wide-area networks, the time delay will depend on the network structure used to send the information to the clients. The type of service offered depends on this delay.

Type of services offered

A commercial data provider for stock exchanges can be of different types, according to the average delivery time (adt) offered for the delivery of the data feeds to the clients:

- *end of day* data provider. Data are delivered at the end of the day
- *delayed* data provider. Data are sent periodically and only when there is a modification.
- *real-time* data provider. Data are sent each time there is a modification.

CSE will satisfy one of these services.

Non-functional requirements: quality characteristics

The quality characteristics required for CSE are the following: - *Availability*, because the system must not interrupt the service. In case of interruption, important transactions may be lost involving substantial financial loss. - *Efficiency*, because the data must be delivered within the established average delivery time (*adt*) in order to fulfill the service

offered. In consequence, high performance must be assured in data transmission. - *Portability*, because the clients which are distributed in different locations, use different development platforms, minimizing the need for changes and adaptations. The programming language used is also involved in this issue.

Availability and efficiency are the most relevant characteristics for CSE.

Efficiency is measured in terms of the number of transactions served each day. It depends on the number of brokers and/or stock exchanges to be served and on the platform used. If more clients are introduced, a hardware with high performance must be considered. Reliability in our case, depends directly on the network (Internet) and the different communication protocols for data transmission; it may affect the availability of the whole system. If the system is not available, the main goal will not be accomplished, hence the system will not conform functionality, so availability is crucial for failure or success. In order to guarantee availability, redundancy of hardware and software must be taken into account and maintenance can also be affected in terms of cost increase. In what follows, a general model for establishing the quality characteristics of real-time monitoring systems will be presented.

ISO 9126 [ISO 98] proposes a generic model, to specify and evaluate the quality of a software product from different perspectives or views, acquisition, development, maintenance. It considers *internal characteristics*, which are related to the software development process and environment and *external characteristics*, observed by the end-user on the final software product. The view of quality, on these bases, can be internal or external, and it is also affected by the stakeholder view in the particular stage of development. An external characteristic can be measured internally, however its name and measure may be different, according to the stage of development. For example, *portability* is an external characteristic according to ISO 9126: we can speak of a portable system, from the point of view of the end-user of the final system. Moreover, the design can be extensible from the point of view of the system engineer in the design phase, we will then speak in terms of *extensibility*. An important issue on software product quality is that the product internal characteristics determine or influence the external characteristics. In order to establish this influence, internal characteristics must be *linked* or related in some way to external characteristics. ISO 9126 define six characteristics that can be subdivided into sub-characteristic, introducing a refinement notion: *Functionality, Reliability, Usability, Efficiency, Maintainability, Portability.* Attributes in the ISO context are the measurable elements of the high level quality characteristics and sub-characteristics.

The generic ISO 9126 model must be customized according to the system's non functional requirements. Figure 1 shows the ISO model adapted to the quality requirements of real-time monitoring systems, considering reliability as the relevant external characteristic. It considers two main aspects: the arrival of the data to their final destination and the correctness of these data at the moment of displaying them on the

client for satisfying the service. In terms of the CSE system, availability is an external sub-characteristic of reliability. If availability cannot be guaranteed, the system is not reliable. Reliability is measured by the percentage of time that the system functions without failures that represent an interruption of the service. Complexity, as its internal sub-characteristic, can be measured registering the interruptions of the system, as the time that the data server is not transmitting the feeds, and the number of clients requiring the services. A great complexity could affect reliability. Coupling is used to calibrate complexity. It is measured in terms of standard OO metrics.

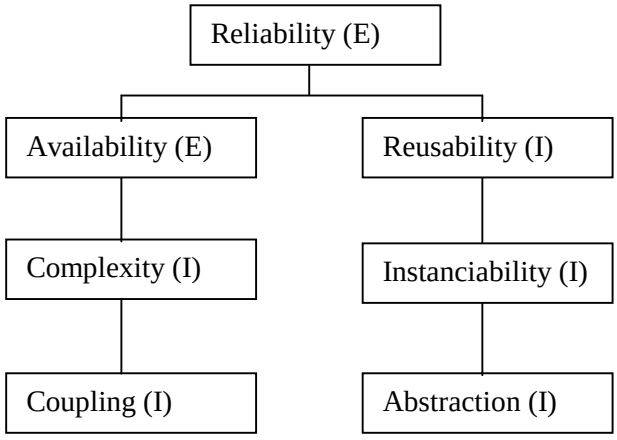


Figure 1. Quality Model for Real-time Monitoring Systems

On the other hand, reusability is an internal sub-characteristic that may also affect reliability. At design level it can be measured using standard OO metrics considering abstraction a sub-characteristics of instanciability.

Efficiency (performance) is an external characteristic measured in terms of the number of transactions served each day. Portability may in turn affect efficiency. They will not be treated here in further details. Usability and Maintainability are not the main concerns for CSE, they neither will be discussed here.

From the above discussion, it can be observed that availability affects directly the functionality or functional conformity of real-time monitoring systems. If the system is not available, the functional requirements will not be fulfilled. In this sense, we have given priority to this characteristic for selecting a convenient architecture for CSE.

2.2 Architectural styles for real-time systems

CSE is a distributed application, so we will be interested only in those architectural styles favoring indirect communication and components decoupling. We will consider the Data Indirection style [KK 99]. This style is characterized by an intermediary (data repository or protocol) between producers and consumers of some shared data. Producers

and consumers do not know the data implementation details of the repository and they do not know each other. The design patterns Publisher/Subscriber and Mediator [Bus et al 96] will be studied. The Data Indirection style describes an elemental distributed software system in which producers and consumers communicate through an intermediary component. However, the details on the repository or the protocol associated to the intermediary component remains undefined.

In order to communicate producers and consumers through a specific communication model, we could introduce variants of the intermediary component. As a result, the Publisher/Subscriber pattern is studied. It introduces the synchronization and propagation of changes between the publisher and the subscribers. The Mediator pattern introduces a specialized component (i.e. Mediator) taking in charge the communication between colleagues which differ in their communication protocols.

3. THE ABAS (Attribute-Based Architectural Styles)

The notion of Attribute-Based Architectural Style (ABAS) [KK 99], as we pointed out in the Introduction, is conceived to make architectural styles the foundation for more precise reasoning about architectural design. This is accomplished associating a *reasoning framework* (quantitative or qualitative) with the *description* of an architectural style. The reasoning framework is based on the establishment of a *quality model* specific to a quality characteristic, called *attribute* in the ABAS approach. Notice that the ABAS attribute notion corresponds to the ISO 9126 notion of quality characteristic. Only one attribute at a time is considered when ABASs are used in design or analysis, because ABAS is associated with only one attribute reasoning framework, called an *attribute model*. For example, if an architectural style is interesting from both a performance and a reliability point of view, it would be motivation for creating the respective performance and reliability ABASs. The authors claim that using ABASs is a step in moving architectural design closer to being an engineering discipline. Design and analysis of software architecture is based on reusable design components: reusing known patterns of software components with predictable properties. The information for characterizing an ABAS quality attribute is divided into three categories: - *External Stimuli* that causes the architecture to respond or change. - *Responses*, that are quantities measured or observed in the requirements or attributes desirable in the architecture. - *Architectural decisions* that are aspects (components and connectors) and their properties, characterizing the style, that have a direct impact on achieving attribute responses. The main purpose of every ABAS is to organize consistently the existing specialized body of knowledge in each of the quality attributes communities. This knowledge can be reused in every ABAS related to a particular quality attribute. Table 1 shows the four parts of the ABAS structure:

This structure is similar to those proposed in the catalogues of architectural styles [SG 95], [Bus et al 96], with respect to Part 1 and 3 of Table 1. The main difference

consists in adding explicitly the information on the characteristics of the quality attribute relevant to the particular style, expressed in Part 2 of Table 1. These are the measures of the responses and constitute the quality model for the attribute. Moreover, Part 4 of the structure, analysis, is used to establish the link between the quality model of the attribute, and the measures of the attribute. The aspects discussed in Parts 2, 3 and 4 constitute the reasoning framework for establishing the quality characteristics of the architectural style.

From the above discussion, an ABAS is seen as a reusable design component, providing a quality model for a specific characteristic which is predictable in the context of the application where the particular ABAS will be used. For example in our case, if the reliability attribute is required, all the ABAS using different forms of data indirection, which seems to be suitable architecture for distributed systems, could be analyzed according to the framework of Table 1. The complexity of the architecture, expressed by the coupling of the components, has to be taken into account, because we are considering explicitly availability. In this sense, we have extended the ABAS framework [CLP 00], considering the ISO 9126 quality model for a global and better understanding of the quality characteristics of the system. The quality model previously discussed, shows how these characteristics affect the availability of the services offered by the system.

Structure	**Description**
1. Problem description	*Informal description of the design and analysis problem* that the ABAS is intended to solve, including the quality attribute of interest or whose presence is desirable in the architectural style, *the context of use*, constraints and relevant attribute-specific requirements.
2. Stimulus/Response attribute measures	A characterization of the stimuli to which the ABAS is to respond and the quality attribute measures of the response. Construction of an ISO 9126 based *quality model for the attribute.*
3. Architectural style	Description of the architectural style in terms of its components, connectors, properties of those components and connectors, and pattern of data and control interactions (their topology) and any constraints on the style. Description of *architectural decisions*.
4. Analysis	Description of how the quality attribute models are formally related to the architectural style and the conclusions about "architectural behavior". *Establishment of the links* or tradeoff, between the quality characteristics required and the measured properties affecting them. A reasoning and analysis and design heuristics are formulated.

Table 1. The ABAS Structure

3.1 Data Indirection

Problem description

This ABAS is characterized by keeping the producers and consumers of shared data from having knowledge of each other's existence and the details of their implementations by interposing an intermediary or protocol between the producer and consumers of shared data items.

Criteria for selecting Data Indirection

It is relevant to anticipate changes in the producers and consumers of data, including the addition of new producers and consumers, if these changes are frequent and it is worth the cost of the modification.

Stimuli/Response for availability

Important stimuli and their measurable controllable responses are:

- Stimuli:
  - add a new producer or consumer of data
  - a modification to an existing producer or consumer of data
  - a modification to the data repository
- Responses:
  - The number of components, interfaces and connections added, deleted and modified, along with the characterization of the complexity of these additions/deletions/modifications

Architectural considerations

The data repository can be a location known by both producers and consumers (e.g. a file or a global data area) or it can be a separate computational component (e.g. a blackboard). The constraint on the repository is that it can hold data. The repository has a data structure, and a set of data types or layout known by all producers and consumers. A single component may be both a producer and a consumer. The producers place their data on the repository because they know the details of the layout; the consumer has a similar behavior for retrieving the data. The management of performance and concurrency control are outside the scope of this style.

Analysis

Redundancy in data producers and data flow channels will increase availability. The dependency on the repository is crucial for availability. In case of failure, a substitute repository must be available.

Architectural parameters for the availability attribute	
Topology	Star
Knowledge of the data layout by client	Complete
Dependency on Repository for producers/consumers	Very high
Redundancy of data producers	High
Redundancy of data flow	High

Table 2. Architectural decisions for Data Indirection

3.2 Mediator

Problem description

Mediator is extensively described in [Gam et al 95], [LL 99]. The intent of the Mediator design pattern is to define an object that encapsulates how a set of objects interacts. Mediator promotes loose coupling by keeping objects from referring to each other explicitly (encapsulation), and let you vary their interaction independently. Consumers and producers are called colleagues. Mediator is a distinguished colleague. It favors the communication among colleagues that do not know each other, but only their Mediator; therefore the number if interconnections is reduced.

Criteria for selecting Mediator

Conditions that must be satisfied to select Mediator:

- Colleagues do not know each other
- A colleague only knows its Mediator
- Mediator knows all its colleagues
- Colleagues are not coupled
- There are no dependency cycles among colleagues
- Mediator is coupled with its colleagues

Stimuli/Responses for availability

- Stimulus: add a new colleague
- Response: availability of the service increases with time, the number of colleagues (relevant to availability of service)

Architectural considerations

Table 3 presents relevant considerations for Mediator with respect to the availability attribute.

Architectural parameters for the availability attribute	
Topology	Star
Size (Number of colleagues)	High
Dependency on Mediator	Very high
Redundancy of data flows	High

Table 3. Architectural decisions for Mediator

Analysis

Colleagues may be data producers or consumers, indistinctly. Redundancy of colleagues implies the capacity of substituting the mediator for another colleague in case of failure, increasing availability as a function of the time that the service is available.

However, if the number of colleagues increases too much (increase in complexity) the capacity of the Mediator for handling communications could be compromised. In consequence, the availability of the system will be negatively affected, since the direct communication between the mediator and its colleagues could be delayed, increasing the possibility of failures in the data delivery.

In case of CSE, the availability characteristic affects the performance of the system, as a function of the cost of the redundancy mechanisms necessary to provide the required availability level, in a convenient time delay.

3.3 Publisher/Subscriber with Push model

Problem description

It helps to synchronize the state of producers (publishers) and consumers (subscribers) of data. When a producer "publishes" a new data, all the subscribers related to the producer, which require the data, are notified and automatically receive the data. In the case of a push model [Bus et al 96], the producer sends data with the notification only to the interested consumers, reducing the number (complexity) of the communications to the consumers and increasing the performance of the application.

Criteria for selecting Publisher/Subscriber with Push model

Conditions that must be satisfied to select Publisher/Subscriber with Push model:

- The number and identity of data producers and/or consumers are not known or may vary
- The temporal ordering between producers and consumers is not known and undergoes frequent changes

- There are no time constraints related to the amount of data that must be produced and/or consumed. There are no synchronization dependencies between the production and consummation of the data.

Stimuli/Responses for availability

- Stimulus: add a new producer
- Response: increases the availability of the service, measured in terms of the number of transactions executed in a unit of time

Architectural considerations

Table 4 presents relevant considerations for Publisher/Subscriber with push model with respect to the availability attribute.

Architectural parameters for the availability attribute	
Topology	Star
Data persistency	Transitory
Size of the data package	Small
Communication Protocol	Selective broadcasting
Dependencies (from producer)	Very high
Redundancy in data flow	High

Table 4. Architectural decisions for Publisher/Subscriber with push model

Analysis

An adequate redundancy of producers and data flow channels decreases the possibility of failures, increasing availability.

The use of the push model with selective broadcasting communication protocols organized in a star topology, favors performance and availability, considering moderate data packages, as a function of the band width of the communication channel and the number of subscribers.

Availability affects the performance of the system, as a function of the cost of the redundancy mechanisms required. The associated computational infrastructure should have enough capacity and support balanced.

4. FORMALIZING ABAS USING B

The ABAS technique has the advantage of supporting a simple and intuitive description of software architectures. It permits to specify the general structure of a software model at a high level of abstraction and to reason about it. It is useful to understand and document systems, allowing a better communication between developers and customers. However, ABAS lacks of precise semantics and remains inadequate to proof correctness and consistency. Therefore, the resulting specifications can be subject to misinterpretations. ABAS is not sufficient enough to develop rigorous applications that require non-functional properties to be ensured.

The approach presented here integrates the B [Abr 96] formal method with the ABAS technique in a complementary manner. The choice of B offers a perfect opportunity to enhance existing semi-formal descriptions of architectural styles. We use the B formal language in order to balance the semantic weakness of ABAS by a rigorous and precise specification. The B formal specification is used to specify precisely the structure, the behavior and also to measure the non-functional characteristics of an architectural style.

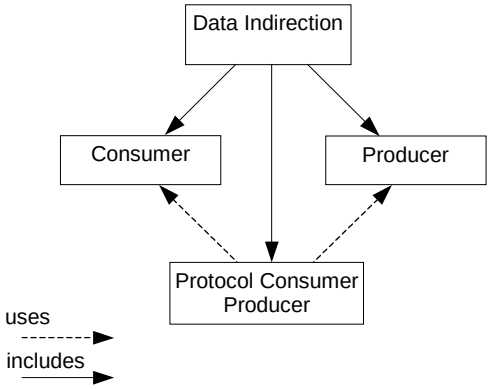
The B formal language is based on the set theory. A B specification is composed of a hierarchy of abstract machines, each one corresponding to a particular component of the specified system. An abstract machine declares a set of state variables describing the abstract state. The machines operations are used to modify the state variables. First order logic is used to express the invariant of a machine, as well as the preconditions of the operations. The post-conditions are defined as generalized substitutions. The consistency between invariants and operations can be proven. The mistakes are consequently removed, ensuring the correctness of the specification. This is a major advantage of the B method. The B method is entirely supported by automated tools such as the Atelier B.

In [MLL 00] we have presented an approach to formally specify architectural patterns using the B language. A complete description of the B method and the B language can be found in [Abr 96].

In the current approach, ABAS technique is used to describe the high-level structure of a style, such as components/classes and association relationships among them. Then, a first B abstract specification is deduced from the ABAS description and used to check consistency. To do so, an abstract machine is associated to each structural component of the style. Subsequently, the B notation is used to describe details of each component that are left unspecified in ABAS, such as the composition and data types of class states, the behavior of class operations and the global invariants. At this level a number of important decisions concerning some unspecified properties must be elucidated by the developer. The quality attributes are included at this point. The resulting specification is then used to determine the quality attribute values of the architecture.

Figure 2 shows the structure of the B specification associated to the Data Indirection style. The left hand side of the picture shows the *uses* and *includes* links between the different machines. On the right hand side, we present the machine *Data Indirection* which specifies the architectural style. It includes the components *Consumer*, *Producer* and *Protocol Consumer Producer* (the intermediary). Because of lack of space, the complete specification of these components is omitted here. In order to measure the non-functional requirements, the quality attributes are associated to the B machines through the *definitions* clauses. Notice that a definition called *availability* is used to associate the availability attribute to the *Data Indirection* machine. The availability of the whole architecture is calculated from the consumer's perspective. For a given consumer (C_j), the availability of the system takes into account five variables :

- the availability of the consumer itself, avail(C_j)
- the availability of the connector, avail(co-I-C_j), between the intermediary and the consumer
- the availability of the intermediary, avail(I)
- the availability of each producer (P_i) communicating with the intermediary, avail(P_i)
- the availability of the connector between each producer and the intermediary, avail(co-P_i-I).




```
MACHINE Data_Indirection
    /* machine specifying the whole architectural style */
INCLUDES Consumer, Producer, Protocol_Consumer_Producer
    /* used machines */
VARIABLES
  message, protocol_consumer_producer, dispatched_messages
    /* state variables of this machine */
INVARIANT
  message ⊆ MESSAGE ∧
  protocol_consumer_producer ∈ consumer +→ producer ∧
  dispatched_messages ∈ message → producer ∧
  ran(protocol_consumer_producer) ⊆ ran(dispatched_messages)
    /* the invariant on variables */
DEFINITIONS
  Availability(Cj) ==
    Σ ( avail(Pi) * avail(co-Pi-I) ) * avail(I) * avail(co-I-Cj ) * avail(Cj)
INITIALISATION
  message,protocol_consumer_producer,dispatched_messages := ∅,∅,∅
    /* the initial state of variables */
OPERATIONS
    /* … operations definitions …*/
```

Figure 2. B specification of Data Indirection style

Figure 3 shows the specification of the Publisher/Subscriber architectural style. Notice that, for given subscriber (S_i), the availability of the system takes into account the following variables:

- the availability of the subscriber itself, avail(S_i)
- the availability of the connector, avail(co-Pub-S_j), between the publisher and the subscriber
- the availability of the publisher, avail(Pub).

As for the Data Indirection specification, the availability attribute of Publisher Subscriber is declared as *definition* within the abstract machine.

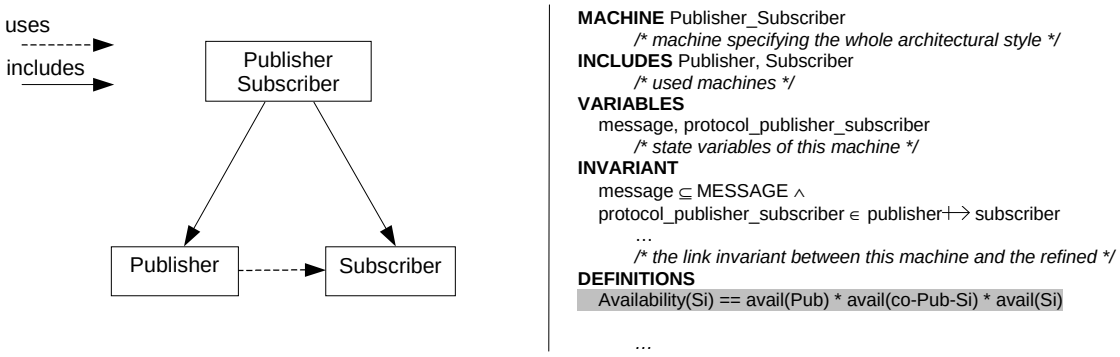


Figure 3. B specification of Publisher Subscriber style

5. Case study: selection of the architecture for CSE.

5.1 Selection of the Publisher/Subscriber with push model for Stock Exchanges Monitoring Systems.

In the previous sections, we have studied different characterizations of architectural models for real-time distributed systems, in particular for stock exchanges monitoring systems. The extended ABAS framework formulated for each candidate architecture has provided useful guidelines for helping in the selection criteria.

- Mediator is not adequate because it favors encapsulation (abstraction) of components (see Figure 1), communicating colleagues that do not know each other by means of an intermediary (Mediator); even if it favors low coupling, it is better adapted for achieving modifiability and reusability, instead of availability.

- Publisher/Subscriber with push model is adequate because it offers a selective broadcasting of the data by the publisher, maintaining at the same time a low level of coupling. The costs of redundancy may be paid, because the structure of the Publisher/Subscriber is not complex. Notice that since all the architectures studied derive from the Data Indirection style, they have in common a high dependence from the intermediary component. Then redundancy is crucial for availability. But if the involved structure of the pattern is simple, complexity will decrease and so will decrease cost.

5.2 Evaluation of the availability attribute

We applied the Publisher/Subscriber style with push model to design the architecture of the CSE. The publisher receives directly the *feeds* from the antenna and broadcasts them to the subscribed brokers via a connector. The brokers are provided with a component subscriber, as shown in figure 4.

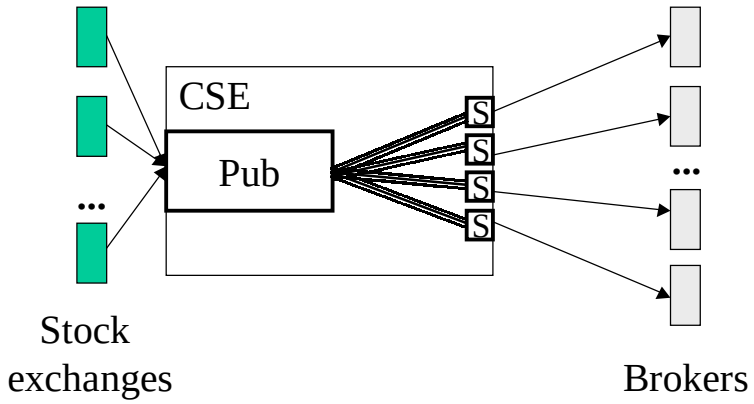


Figure 4. Application of Publisher/Subscriber style

The availability attribute for this architecture and for the i^{th} broker is the following:

$$avail_i = avail(\ Pub\ ) * \ avail(co\text{-}Pub\text{-}s_i\ ) * avail(\ S_i)$$

where avail(Pub) is the value of the availability attribute associated to the publisher machine, avail(S_i) is the one of the subscriber machine and avail(co-Pub-s_i) the one of the connector between the publisher and the i^{th} broker.

In order to enhance this availability, we choose to use Internet as a connector. Internet can be considered as always available, i.e. its availability is equal to 1. The architecture obtained is shown in Figure 5.

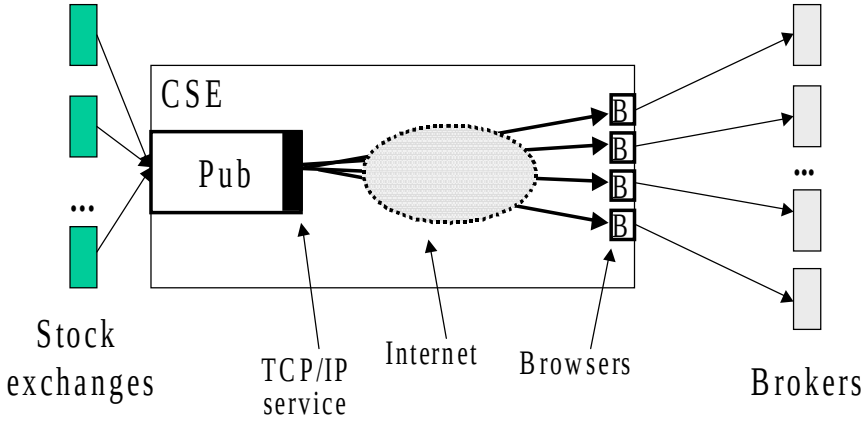


Figure 5. Use of Internet as connector

The availability attribute for this architecture and for the i^{th} broker is now the following:

$$avail_i = avail(\ Pub\ ) * avail(\ B_i)$$

where avail(B_i) is the availability of the browser used by the broker to interact with the publisher.

This availability formula shows that the availability of the publisher is crucial. The introduction of a redundant publisher will double this availability. The architecture obtained is shown in Figure 6.

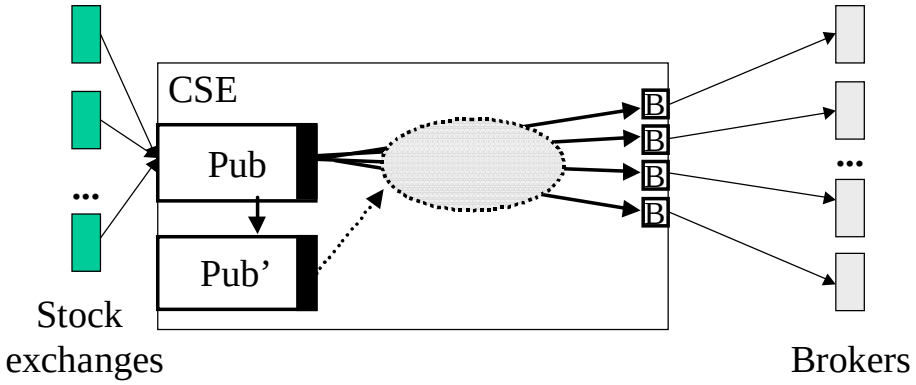


Figure 6. Introduction of a redundant publisher

The availability attribute for this architecture and for the i^{th} broker is now the following:

$$avail_i = 2 * avail(Pub) * avail(B_i)$$

6. CONCLUSION

In this paper we have studied different attribute-based architectural styles (ABAS). The styles have then been formalised using the B language. Each component is specified as an abstract machine in which quality attributes are defined. We have taken the availability attribute as an example. Then, we have applied the proposed technique in order to design the architecture of a Stock Exchanges Monitoring System. The architecture has been developed by stepwise transformations: first we have used the Publisher/Subscriber style with the push model. The formula of the availability attribute showed the importance of the connector's availability. The use of Internet as connector between the publisher and the subscribers had the advantage to offer a very high availability. Then it appears that the availability depend on the availability of the publisher itself. The middleware solution consisting in introducing a redundant publisher was then applied.

The correct selection of a system architecture enhances the subsequent software implementation and the system as a whole. Moreover, the structural characteristics or topology of the chosen styles influences the overall quality goals. However, the applicability of a style, that is to say the selection of the right style for a particular design issue, is yet an open problem. It has been the object of many relevant works [Gam et al 95], [Bus et al 96], trying to describe patterns to be easily retrieved and reused. However, these attempts lack in general of a standard and formal notation, being limited to an informal description and examples of the application of a style. Quality issues are not

explicitly considered. Therefore, this descriptions lead to misinterpretations. This makes them an insecure basis for critical software development.

Formal methods are used to specify precisely the structure and the behavior of the entities composing a system and to prove rigorously that these satisfy the desired structural and behavioral properties. Formal methods promise increased reliability of software systems and provide analysis and verification tools. In [MLL 00], we have introduced a formal framework for system development using patterns. This framework integrates the B formal language, describing the transformation from software architecture to system design through successive transformation steps. In this paper we have shown how formal methods can also take into account quality attributes.

REFERENCES

[Abr 96] Abrial J.R. "The B Book - Assigning Programs to Meanings", Cambridge University Press, 1996. ISBN 0-521-4961-5.

[BCK 98] L. Bass, P. Clements, R. Kazman "Software Architecture in Practice", Addison Wesley, 1998.

[BK 99] L. Bass, R. Kazman "Architecture-Based Development", TR CMU/SEI-99-TR-007, ESC-TR-99-007, April 1999.

[Bosh] J. Bosh "Design and Use of Software Architecture", ACM Press, 2000.

[Bøe et al 99] Bøegh J., DePanfilis S., Kitchenham B., Pasquini A. "A Method for Software Quality Planning, Control and Evaluation". IEEE Software, 69-77, March/April 1999

[Bus et al 96] F. Buschman et al "Pattern-Oriented Software Architecture. A System of Patterns", John Wiley & Sons Inc., 1996.

[CLP 00] Chirinos L., Losavio F., Pérez M.A. "Attribute-Based Techniques to Evaluate Architectural Styles for Interactive Systems", Centro ISYS, Universidad Central de Venezuela, Caracas, May 2000, Draft.

[Dou 99] Douglass B. P. "Real-Time UML" Second Edition, Addison-Wesley, 1999.

[Gam et al 95] E. Gamma, R. Helm, R. Johnson and J.Vlissides "Design Patterns – Element of Reusable Object-Oriented Software". Addison Wesley, New York 1995.

[ISO 98] ISO/IEC FCD 9126-1.2: "Information Technology - Software Product Quality.Part 1": Quality Model, 1998.

[KK 99] Klein M., Kazman R., "Attribute-Based Architectural Styles", CMU/SEI-99-TR-022, ESC-TR-99-022, October 1999.

[Kru 00] P. Krutchen "The Rational Unified Process. An Introduction", 2nd. Edition, Addison Wesley, Reading, Massachussets, 2000.

[Kaz et al 98] Kazman R., Klein M., Barbacci M., Longstaff T., Lipson H., Carriere J., "The Architecture Tradeoff Analysis Method",CMU/SEI-98-TR-008, ESC-TR-98-008, July 1998.

[LC 99] Losavio F., Chirinos L. “Evaluación de la calidad en el desarrollo de sistemas interactivos”, (92-108) Proceedings X CITS, Curitiba, Brazil, 17-21 May, 1999.

[MLL 00] Marcano R., Lévy N., Losavio F. “Spécification et Spécialisation de Patterns en UML et B”. Proceedings LMO’2000 – Langages et Modèles à Objets, Ed. Hermès, Montréal (Ca), janvier 2000.

[SG 96] Shaw M., Garlan D. “Software Architecture – Pperspective of an Emerging Discipline”, Perentice Hall, 1996.

[SR 98] B. Selic, J. Rumbaugh “Using UML for Modelling Complex Real Time Systems”, RSC, OTL, March 1998.

QWE2000 Session 2M

Mr. Kie Liang Tan [Netherlands]
(CMG TestFrame Finance)

"How To Manage Outsourcing Of Test Activities"

Key Points

- Advances in quality control process
- Risk management
- Improving software process

Presentation Abstract

"Outsourcing" is a process in which a company transfers all or part of one of its departments to an outside vendor who then handles the company's affairs for a price that is spelled out in the outsourcing contract.

Test Managed Services (TMS) is a method to managed test activities, which is out sourced by a company to the vendor (is called the TMS service-center).

The test activities are:
* Managing the testware (test scripts, test cases, test database and documentation),
* Managing the test environment (test infra-structure, i.e.: hardware, software, test tools),
* Test preparation, execution and reporting
* Support in test consultancy

In this paper we describe various outsourcing projects to managed services related to the maintenance of application software and the test activities. The test managed services is an important part of the software development and verification process to ensure for good quality and good time-to-market of the application software.

The author will present the method to prepare, implement and execute this Test Managed Services.

About the Speaker

The author has studied (1971 - 1977) in the Technical University of Delft as Electrical Engineer with field of specification Information Technology and Data Communication. The author has more than 20 years experience in ICT (Information Communication Technology) in different area's, i.e: Telecommunication, Medical Systems, Space

Systems and Finance. The authors has significant experience in the area of test and verification coordination and management. The author has joined CMG since 1998 and worked in the Division Testmanagement & consultancy as senior management consultant.

How to manage outsourcing of test activities

Kie Liang Tan
CMG – TestFrame Finance
kie.liang.tan@cmg.nl

Agenda

- Test managed services (TMS)
- TMS Categories
- TMS phases:
- Classical problems of a tester
- Recommendation

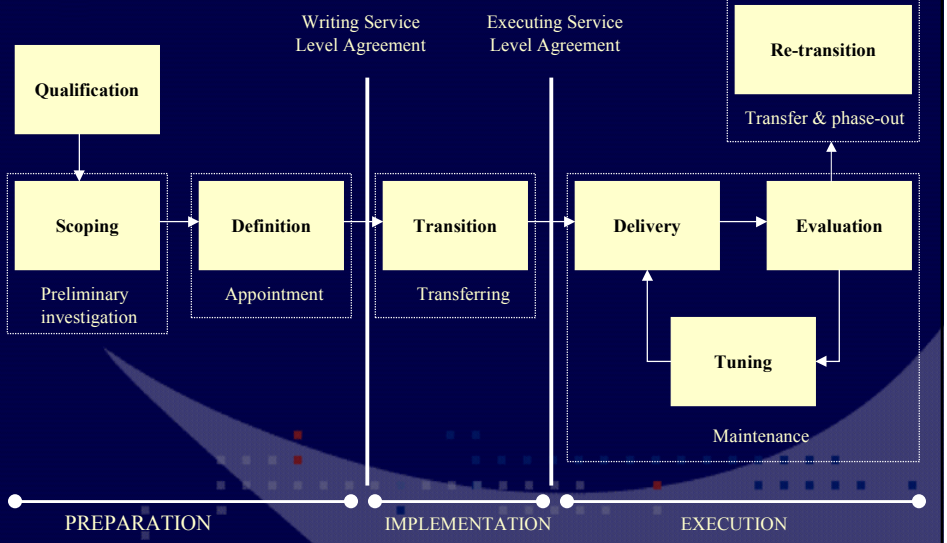
Test managed services

- Test managed services are services in the area:
  - maintenance of testware
    - navigation scripts,
    - clusters,
    - test data and
    - test documentation
  - maintenance of test environment
    - hardware, software and test tools
  - test preparation,
  - test execution and
  - test reporting

Test managed services categories

- outsourcing: transfer of all activities from a client to a TMS test-centre
- insourcing: the services are performed at client's location
- smartsourcing: set-up the test organisation in close co-operation between the client and the test-centre, and select between outsourcing en insourcing

Phases: Test Managed Services



Qualification phase

- Interviews with the client to investigate:
  - client motivation to oursource
  - cost benefit determination
    - what are the services
    - what are the benefits for the client

Scoping phase

- Analyse and investigate
  - the application to be tested
  - the test strategy and test organisation
  - the test environment and testware
- Investigate the current and desired situation
- Analyse the risks of the project
- Scoping report

Definition phase

- Define and agree with client:
  - what has to be done, such as:
    - basic service level
    - test preparation and execution
    - maintenance testware
    - test facilicities and hosting
  - where the services will take place
  - when the service will be performed
  - who will do what

Transition phase

- Knowledge transfers
  - bussiness knowledge and application
  - test environment and testware
- prepare service related procedures
- setting-up service related organisation
- writing test plan
- perform acceptance test

Delivery phase

- Performing the service level request triggered by changes in:
  - application to be tested (release note)
  - the test environment

Evaluation phase

- Evaluate the service:
  - on strategic level: contracts and cost
  - on tactical level: required adaptation of the service and reporting
  - on operational level: workmethod, process and test information

Tuning phase

- change the service to improve it using the result of the evaluation :
  - operational level: is a continuing process
  - tactical level: is a major process change
  - strategic level: the scoping and definition phases have to be performed again.

The classical problems of a tester

- Testteam is not properly inform on the function/ design changes
- Release note (documentation) is incomplete
- Not enough time for test: software is late and implementation date is fixed.
- The quality of the application software is poor
- The users are unsatisfied: the application is not properly tested.

Recommendations

- Combine the maintenance of the application software and the test managed services into one responsible company

or

- Put extra attention on the qualification, scoping and definition phase to cover the conditions in the Service Level Agreement.

How to manage outsourcing of test activities

Test Managed Services (TMS)

This paper describes the reference model and the process of Test Managed Services, which give services in the following areas:

- Maintenance of testware (navigation scripts, clusters, test database and documentation),
- Maintenance of test environment (test infra-structure, i.e.: hardware, software, test tools),
- Test preparation, execution and reporting.

This paper is based on our experience in various projects related to Test Managed Services.
The target audience for this paper/presentation is:

- o Application Oriented
- o Management Oriented
- o Technical or Technology Related

TMS categories

TMS can be distinguished by three categories:

1. Outsourcing: the client transfers the maintenance of the test environment and the testware to the TMS service-centre location. The test preparation and execution are performed by the TMS service-centre as well. For example: a publisher in Holland is outsourcing the test automation and execution activitities to our TMS service-centre.
2. Insourcing: the activities will be performed at the client's location, for example: at a Dutch government office, we set up an internal test-centre
3. Smartsourcing: the activities will be performed in close co-operation between the client and the TMS service-centre which, depending on the situation can be a mixed of outsourcing and insourcing. For example: for a large bank we helped them to set-up the test environment and the test organisation, and then made the decision to perform outsourcing or insourcing.

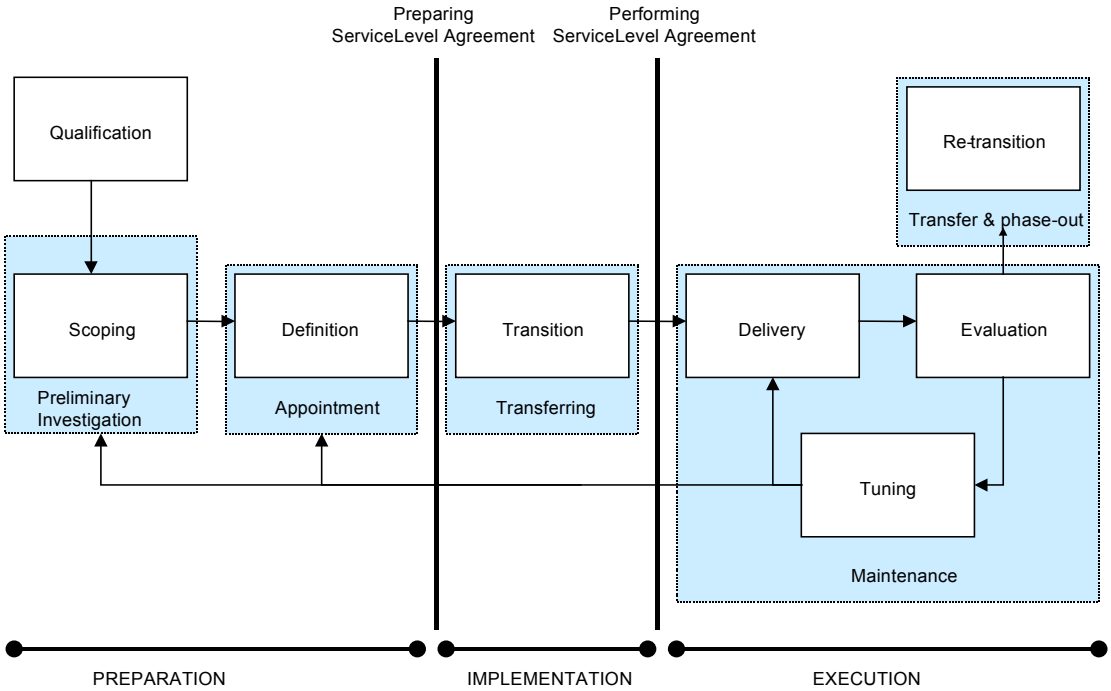
TMS phases

TMS life-cycle can be divided into eight phases (see figure), i.e.:

- Qualification
- Scoping
- Definition
- Transition
- Delivery
- Evaluation
- Tuning
- Retransition

Each of the phases is described below:

Test Managed Services phases



Qualification
This phase is basically a cost/ benefit determination; the TMS service-centre is discussing with the client what are the services and what are the benefits of the TMS. Interviews will be held with the client to find out the bussiness triggers and the client's motivation for transferring the test maintenance activities to a professional TMS service-centre.

Scoping
In this phase the TMS service-centre will analyse the application to be tested, the test-strategy, the testware, the test environment and test organisation. The analysis includes also interviews with the client and available documentation.
The findings, conclusions and recommendations will be described in the Scoping-report, as follows:

(1) The client's test organisation and the role of the TMS service-centre to support the test knowledge,
(2) Investigation of the current- and the desired-situation: manual or automatic tests, test execution and reporting,
(3) Defining the test strategy
(4) The type of the test to be maintained: System test, Functional Acceptance Test, User's Acceptance Test or Production Acceptance Test,
(5) The configuration of the testware and test environment, and the maintenance-approach,
(6) Risk analysis of the project, i.e.: the system to be tested, the test environment, the testware and the organisation.

(7) The status of the documents, handbooks, procedures, etc.
(8) The release-frequency of the application to be tested and the types of changes.

Definition
The TMS service levels are planned and discussed with the cliënt and this will result in an Service Level Agreement (SLA).

In the SLA the following is established:

- What has to be done
- Where the services will take place
- When the services will be performed
- Who will do what

The TMS services, stated in the SLA, will be written in an quotation together with the findings in the Scoping report. After agreement is reached about the quotation, then the service contract will be made containing:

- The sevice level of the current maintenance organisation,
- The desired service level of TMS with a step-by-step plan for implementation,
- The required manpower in number, knowledge and experience.

Transition
In this phase the following activities will be performed:

- knowledge transfer to the TMS service-centre of all items related to test activities depending on the agreed service levels, i.e.:
  - ⇒ bussiness knowledge
  - ⇒ application software to be tested knowledge
  - ⇒ testware and test data
  - ⇒ test environment
  - ⇒ test strategy
- setting-up TMS related procedures such as: ITIL procedures (configuration-, change-, problem-, service-level-management),
- writing test plan,
- setting-up TMS related cliënt organisation,
- perform acceptance tests to verify and accept the testware, test environment and the related application software to be tested,
- After evaluation of the acceptance tests, the cliënt and the TMS service-centre can establish that the maintainance can be taken over by the TMS service-centre in a controlled manner.

Delivery
This is the phase to perform the service as defined in the Service Level Agreement (SLA).
The service (i.e.: prepare test environment, testware and test data, execute test and archive new testware) can be triggered by the following request:

- Release-note of the application to be tested containing the information about the new

functions/ modification in the application, the planning when the application will be ready and the relevant documentation (such as: functional design).

- Release-note of the test environment containing the information about the changes in the test environment, the planning and the relevant documentation.
- This request must be submitted on time so that the TMS service-centre will have enough time to prepare the test and execute the test as soon as the new application to be tested or the test environment is ready.

Evaluation

In order to ensure that the TMS services comply to the wishes of the cliënt, a total TMS evaluation will be performed periodically on three levels, i.e.:

- Strategic (the Service Contract): partnership, contracts, new services and cost.
- Tactical (the Service Levels): service level evaluation, does the current TMS service comply to the agreement, are there trends detected (incidents, change request, growth) requiring adaptation of the services, reporting and the frequency of reporting.
- Operational (Procedures): small changes in the workmethod/ process, test information and technical tuning.

Tuning

As a result of the previous phase, the TMS service might have to be changed.
These changes can vary between small changes and new kind of services.
In the figure the different changes are indicated by arrows.
The change request for “execution” (Delivery/Evaluation/Tuning) is coming from the operational level; this is a continuing process.
Major process changes will come from the tactical level.
Changes from the strategic level may have impact on the phases ” Scoping en Definition” , so that these phases must be performed again.

Re-transition

At the end of the contract depending on situation and agreement with the client; the testware, the test environment and the documentation might have to be handed over to the client.

QWE2000 Vendor Technical Presentation VT2

Lisa Hoven
(Rational)

"Successful Testing Through Requirements Management"

Key Points

- Function Testing
- Requirements Management

Presentation Abstract

Abstract to be supplied.

About the Speaker

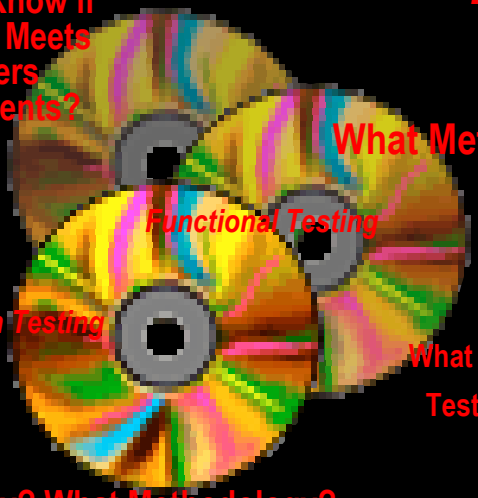
Ms. Hoven is a Technical Consultant with Rational Software BV where she specializes in Testing methodologies and Solutions Consultation, Product Management and product integration. She has been involved in Application Life Cycle development for the past 16 years for companies worldwide. Rounding out her career, Ms. Hoven also has many years of experience in International Sales Support, Application Life Cycle Management Consultant, Business Analysis, Applications Developer, and Systems Programmer. Originally from New Haven, Connecticut USA, she is a permanent resident of the Netherlands where she has lived for the past 2 ½ years.

Agenda

- Development Process
- Iterative Development Life Cycle
- Testing Life Cycle
- Planing, Designing and Implementing Tests
- Manual and Automated Testing
- Evaluation
- How To's
- Q&A

Rational
the e-development company

"Here is my build. How do I test it?"



How Do I Know if This Build Meets the Users Requirements?

Load Testing

What Metrics do I Keep?

Functional Testing

Regression Testing

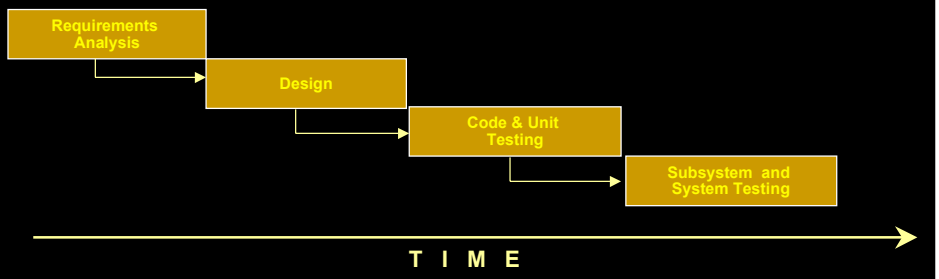
What Will Automated Testing do for me?

Methodology? What Methodology?

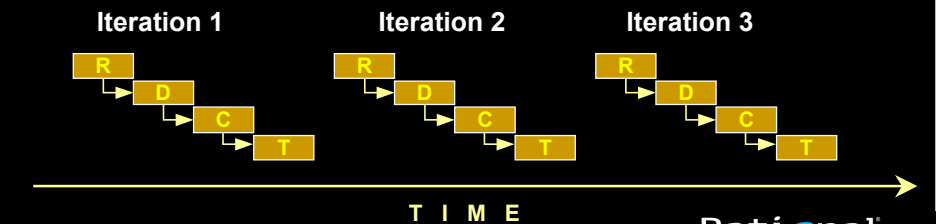
Rational the e-development company

Development Process

Waterfall Method

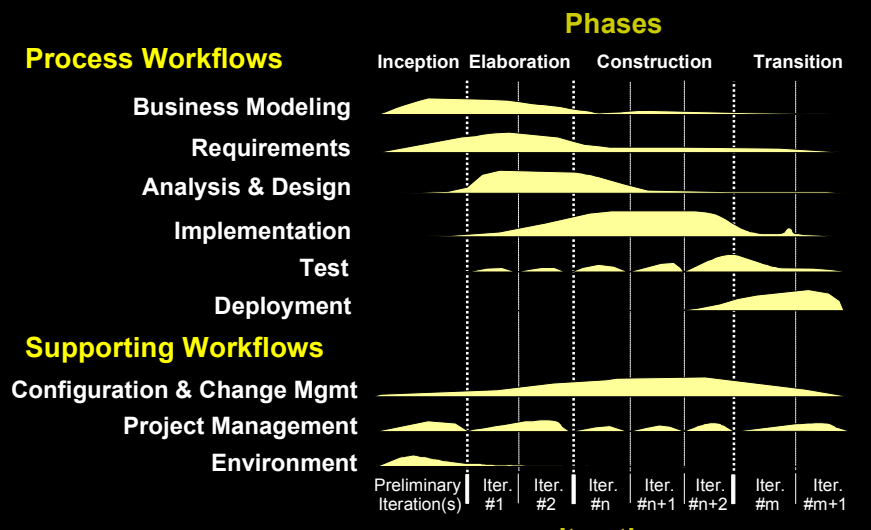


Iterative Process



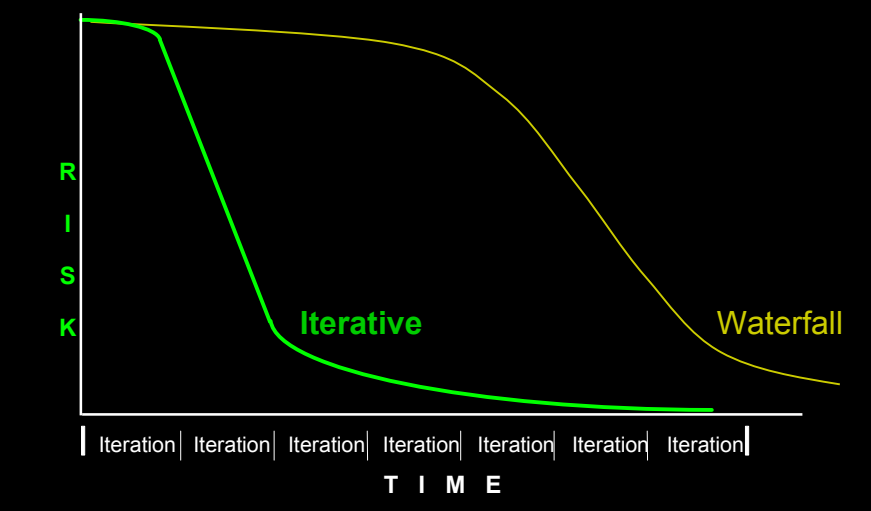
Rational the e-development company

Apply Best Practices Throughout the Life Cycle



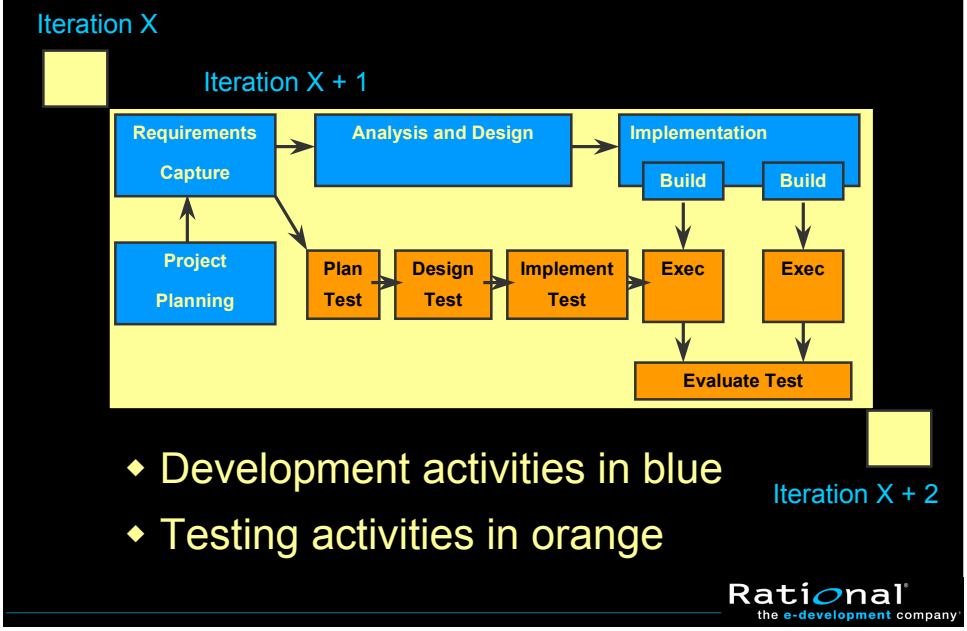
Rational the e-development company

Development Process – Reducing Risk



Rational the e-development company

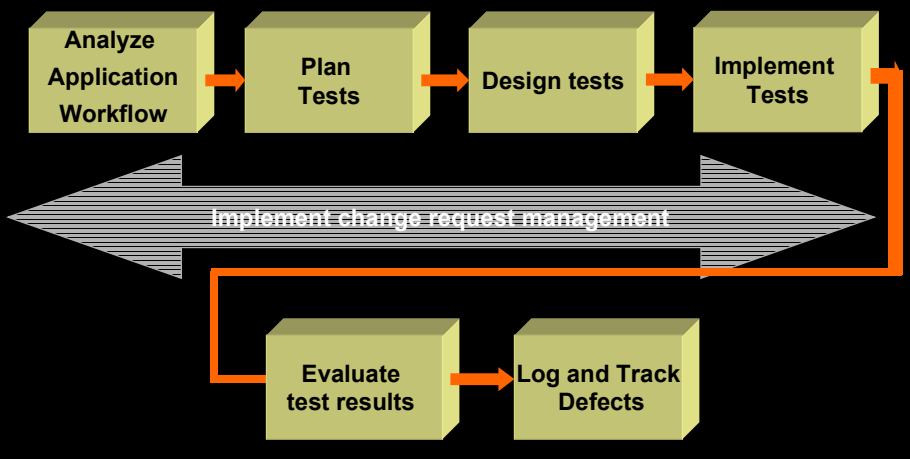
The Test Life-Cycle in a Iterative Approach



- Development activities in blue
- Testing activities in orange

Rational the e-development company

How Do I Implement Good Testing Practices?



Rational the e-development company

Analyze the Application Workflow

- Walk through use cases/business requirements
- Look for patterns in navigation
- Identify specific test cases with data to be used
- Diagram, model, or list the sequence of events
- Determine which procedures are candidates for automation
- Determine areas and functions that will need to be baselined for regression testing.

Rational
the e-development company

Plan Tests

- Identify Testable units
  - Identify Existing Process
  - Define Logical and Physical flows for Process
  - Identify Dependencies
  - Identify Manual Processes
- Create Test Plan

Components of a Test Plan

- Introduction
- Test requirements hierarchy
- Test strategy
  - Test objective
  - Test tools
  - Test techniques
  - Test metrics
  - Completion criteria
  - Special considerations
- Schedule of resources and roles
- Project schedule and milestones
- Schedule of deliverables

Rational the e-development company

Designing Tests

- Objectives
  - Identify/define test cases and test procedures
  - Generate test model
- Inputs
  - Test plan
  - Use-case model, specs
  - Design model, tech specs
- Outputs
  - Test model
  - Test cases
  - Test procedures
  - Test data requirements

Rational the e-development company

Test Cases

- Reflect the requirements for test
- Provide a means to verify the use cases and requirements
- Form the foundation of test design and test effort

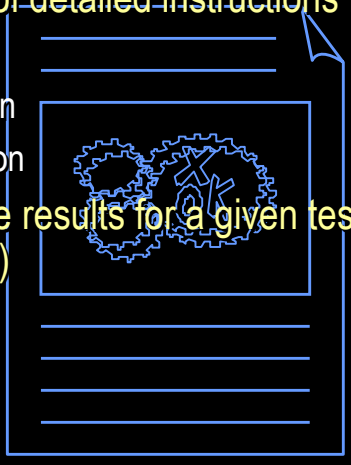
Test Type - Functional

TC ID #	Condition	PIN	Acct #	Amount	Result
CWI	Withdraw pre-set amount	Valid 3112	Valid 5600184	$50.00	Successful cash withdrawal

Rational
the e-development company

Structure Test Procedures

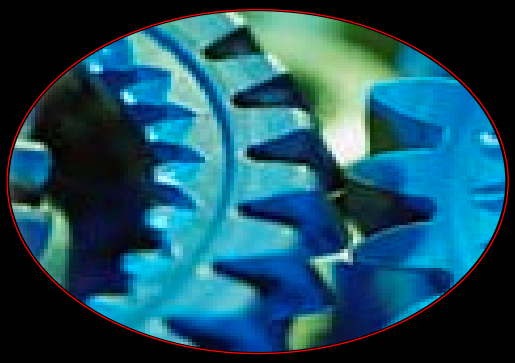
- Are a set of detailed instructions for:
  - Setup
  - Execution
  - Evaluation
- Include the results for a given test case (or set of test cases)



Rational
the e-development company

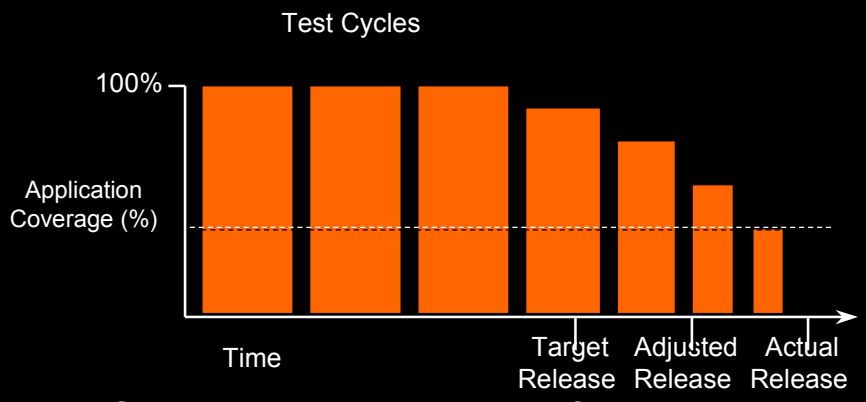
Implementation of Tests

- Manual Testing
- Automated Testing



Rational
the e-development company

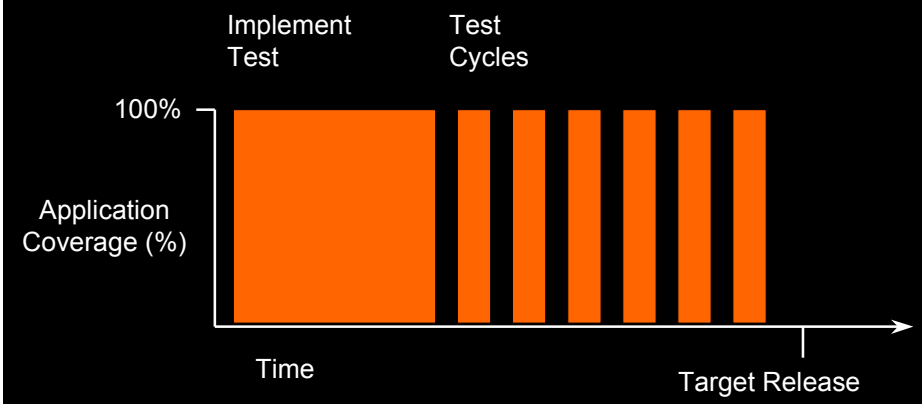
Implementation: Manual Testing



- QA can't match the speed of development
- QA can't fully test each build
- As time runs out, test coverage decreases and risk increases

Rational
the e-development company

Implementation: Automated Testing



- Increased coverage and decreased risk
- Automated test cycles fully test each build
- Frequent and comprehensive tests ensure quality

Rational the e-development company

Automated Testing

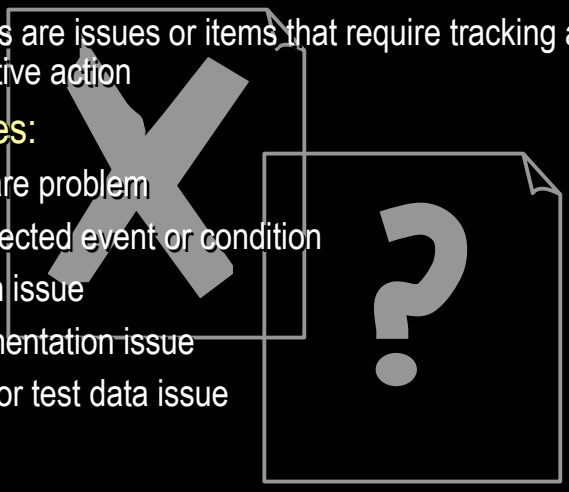
Automated Testing is NOT a Silver Bullet!!

- Keep Proper Expectations for Your Testing Tools.
- Test New Processes Manually and Automate the Existing Processes.

Rational the e-development company

Evaluation: Defects

- Definition:
  - Defects are issues or items that require tracking and/or corrective action
- Examples:
  - Software problem
  - Unexpected event or condition
  - Design issue
  - Documentation issue
  - Script or test data issue



Rational the e-development company

Evaluation: Defects as Metrics

Don't Let a Date Be Your Release Criteria!!!

- Use Quality Metrics.
  - Defects Per "X" Lines of Code
  - Defects Per "X" Requirements
- Defect Trending
  - See Quality Improve Build by Build!
- Communicate
  - Addition of new defects
  - Modification of existing defects

Rational the e-development company

In a Less than Perfect World…..

- What Can I do if Requirements are Not Available to me?
  - How can I Re-engineer these Requirements?
    - Functional/ Technical WalkThroughs
    - Sign-off
- How do I manage Changing Requirements?
  - How can I manage Requirement Impact Analysis?
- How Can I Make Steps to Improve My Process from the Start?
- How Can I Get the other members of the Team to buy into our new-and-improved process?

Rational the e-development company

What Can I do if Requirements are Not Available to me?

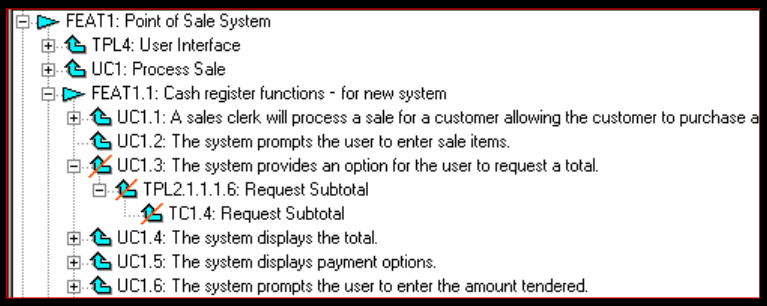
- How can I Re-engineer these Requirements?
  - Functional / Technical Walkthroughs
    - With Users
    - With Analysts
    - With Developers
  - Get Sign-off
  - Assume no Responsibility until Signed Off.

KNOW WHAT YOU'RE TESTING!!!

Rational the e-development company

How do I manage Changing Requirements?

- How can I manage Requirement Impact Analysis?



- How Do I Communicate These Changes ?
  - Keep Requirements Accessible to Everyone
  - Diverse Teams: Liaison Groups

Rational the e-development company

How Can I Make Steps to Improve My Process from the Start?

- Reverse Engineering Function/Feature Requirements from Testing Requirements.
- Adopt a Methodology
  - UML / Rational Unified Process
  - T-MAP
  - TestFrame
- Plan for Planning
  - Creating appropriate documents
  - Staffing
  - Time/Project management

Rational the e-development company

Getting Buy-In From Team Members

- Make Small Changes Iteratively
- Do Not use Force!
- Be the Example
- Make your quality efforts known to other areas of your application that are not under your control.

Rational
the e-development company

Summary: Your Lessons For Today

- Develop Iteratively
- Test From Requirements
- PLAN, PLAN, PLAN
- Set Realistic Expectations for Automated Testing

Rational
the e-development company


speed
quality
Rational®
the e-development company™
Thank You
Rational®
the e-development company™

QWE2000 Session 3T

Dr. Antonia Bertolino, F. Basanieri [Italy]
(CNR-IEI)

"A Practical Approach to UML-based Derivation of Integration Tests"

Key Points

- Deriving tests from UML descriptions
- Integration/System testing
- Use-Interaction Test

Presentation Abstract

In recent years, the widespread use of the object oriented (OO) paradigm has given rise to the need of defining suitable models for describing and analysing the characteristics of complex systems designed according to it. OO system models come usually accompanied by graphical notations, so that the relationships between the elements forming the system can be easily visualized.

UML, the Unified Modeling Language, is the emerging graphical notation to model, document and specify OO systems along all the phases of the software process. Indeed, there exist now many studies about using UML for design, and many tools are available. However, only a little part of these studies so far has addressed the usage of UML for testing, and none of the available commercial tools provide specific assistance for test planning from the UML descriptions.

In this paper we address UML-based testing. Some authors have proposed methods to translate an UML description into another formal description, and then derive the tests from the latter (e.g., [1]). This is not our goal: we aim at a test method that is entirely based on UML, so that it can be easily adopted by industries already using UML. Some other recent studies propose methods to automatically derive test cases from UML statecharts (e.g., [2], [3]). These studies are interesting, and we see them as complementary to ours, as we do not use statechart diagrams, but higher level descriptions of the system. Indeed, we want to address test planning for the integration test phase starting from the very first stages of system design.

We present an approach for UML-based test planning, that is called Use-Interaction testing, as it mainly uses the UML Use Case and Interaction diagrams (specifically, the Message Sequence diagram). Indeed, integration testing is aimed at verifying that the pre-tested system components interact correctly. UML Interaction diagrams can provide the information of how the system components should interact.

The proposed methodology is very simple, and is inspired at large by the well-known Category Partition method [4]: we first look at the Use Case diagram to identify the suitable steps of an incremental bottom-up

test strategy. For each identified sub-Use Case, we then look at the Message Sequence diagram to identify the relevant components, or "Test Units". For each Test Unit, we derive the relevant "Settings" (could be parameters, variables, environmental states) and the relevant "Interactions" (essentially, messages from other Test Units), and identify for them the significant "Choices" (with the same meaning of the Category Partition method). To do this, we can analyse the related Class Diagrams, as well as other design documentation. Hence, by following the sequences of messages between the components over the Message Sequence Diagram, we construct the Test Cases, whereby each Test Case is characterized by a combination of all suitable choices of the involved Settings and Interactions.

We have applied the method to a case study, Argo/UML [5]. Argo/UML is an open source tool that provide cognitive support to system design. We have developed quite a few Test Cases, and we have been able to identify a few bugs in the case study. However, the method is currently manual, and still under evaluation. Our short term plan is the realization of a tool that support it, by smoothly expanding an existing UML design tool.

In the paper we will describe the method in more detail, through examples on the Argo/UML tool, and also will provide a UML definition of the method itself by means of the UML extension mechanism of stereotypes.

About the Speaker

Antonia Bertolino graduated (Laurea degree) cum laude in Electronic Engineering at the University of Pisa in 1985. Since 1986, she has been a researcher with the Institute for Information Processing of the Italian National Research Council (CNR), in Pisa, where she currently heads the Research Department on "Methods and Tools for Software Systems". Her research interests are in software engineering, especially testing, and in software dependability. Currently she is investigating approaches for rigorous integration test strategies and for improving the cost/effectiveness of testing techniques, as well as methods for the evaluation of software reliability. On these topics, she also likes to stay in touch with industries.

Since 1994, she is an Associate Editor of the Journal of Systems and Software, North-Holland, and recently has also joined the Editorial Board of the IEEE Transactions on Software Engineering. She has been the General Chair of the Second International Conference "Achieving Quality in Software AquIS '93", held in Venice (Italy) in October 93. She has been member of the International Programme Committees of several conferences and symposia, including ISSTA, Joint ESEC-FSE, ICSE, SEKE, Safecomp and Quality Week. She is currently Key Area Specialist for the Software Testing Knowledge Area of the Stone Man phase of the ACM/IEEE project Guide to the SWEBOK (Software Engineering Body of Knowledge). She has published over 40 papers in international journals and conferences.

A Practical Approach to UML-based derivation of integration tests

F. Basanieri, A. Bertolino

IEI-CNR (Pisa, Italy)

Contents:

- Integration testing and Software Architecture
- The UML standard notation
- The Use-Interaction Test Approach
- The case study
- Conclusions and Future work

Integration Testing:

Testing interactions (only) between system components (processes, packages, subsystems...) at an appropriate level of abstraction

(handling CBSE and COTS)

SA = Structure + Behavior
(statics) & (dynamics)

- **Integration Testing +**
- **Software Architecture =**

Architecture-based Testing

Envisaged benefits of SA-based testing:

- Right level of abstraction (not too much, not too little)
- Explicit representations of early design decisions (hardest to change)
- Enhances test reusability
- Provides a context for component-based testing
- Confinement within a specific architectural style (patterns for testing)
- Rigorous conceptual basis for the testing of non-functional properties

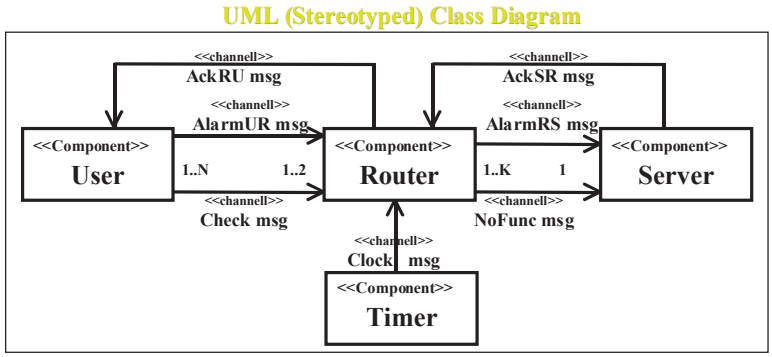
Combining Structural- and Specification-based Test Approaches

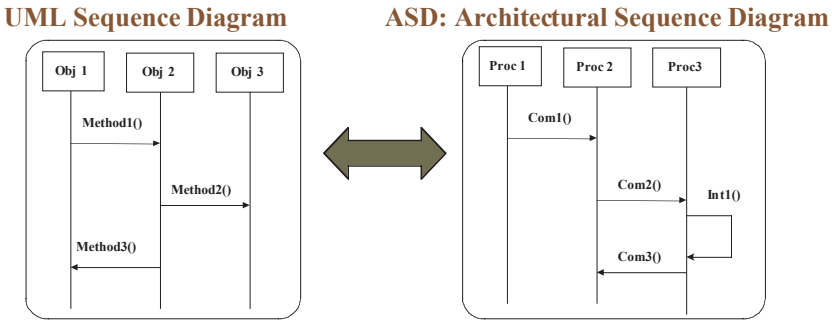
Structural Testing	Specification Testing
✧ Based on code coverage ✧ Modification to code -> new tests ✧ Too complex for real systems ✧ Too syntactic oriented and not semantic (behavioral) ✧ Only late, Code level testing	✧ Needs formal specifications (SDL, CCS, LOTOS, Estelle, Z…) ✧ Based on FSM, LTS ✧ Automated Test Plans ✧ Only for small systems ✧ Specification description too complex and error prone

UML Unified Modeling Language

- Modern complex distributed systems require appropriate models for analysis and design
- UML: standard graphical notation to model, document and specify OO system along all the phases of the software process
- UML uses:
  - views
  - diagrams
  - exthension mechanisms
- Not much used (so far) to guide the testing phases

UML & SA:





Use Interaction Test methodology

a method to systematically derive Integration Test suites from UML diagrams

ENTIRELY UML-BASED (no other language required)

Mainly used UML diagrams:

- **Use case diagrams:** represent a specific functionality provided by the system.
- **Sequence (Interaction) diagrams:** show the dynamic collaborations between a certain set of objects via message exchanges.
- **Class diagrams:** summarize parameters and inputs for the tests

Use Interaction Test methodology

- Incremental test strategy (over the Use Case organizational structure)
- Inspired at large by *Category Partition* method:

  it defines classes of functional equivalence selecting significant values (choices) of relevant categories (parameters or data)

- *Seven steps involved*

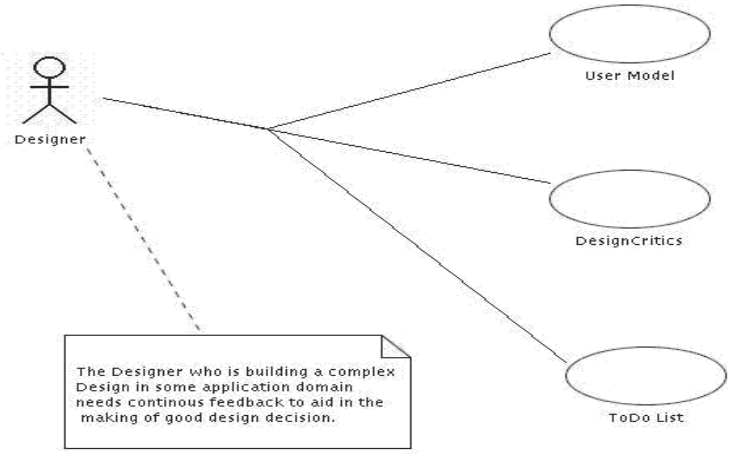
Case study: Argo/UML

Open source tool, supporting UML OO design .

Cognitive support to design (Kernel package):

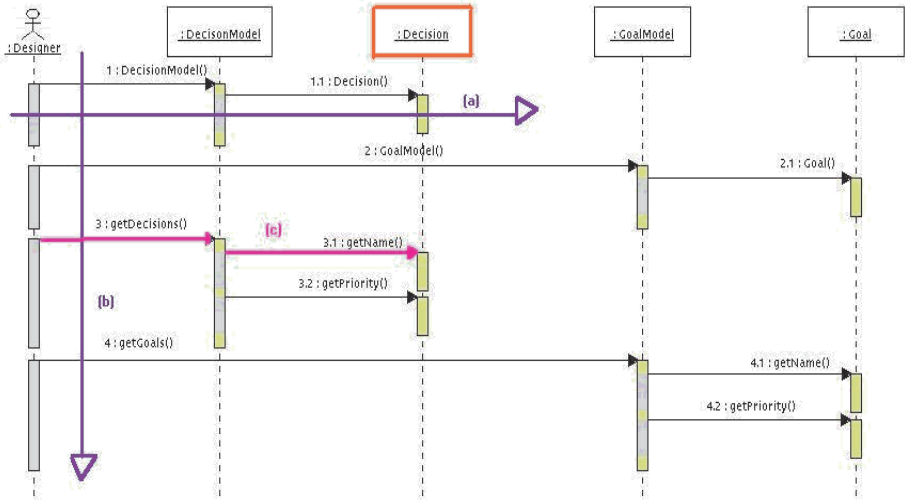
- **User model:** mantains information on designer's choices about project and its features.
- **Design Critics:** critics about building design.
- **ToDo List:** items helping designer about many details of his work.
- **Checklist:** used at the end of design to remind designer to cover all design details and avoid common errors.

Step 1:UML design analysis and search of relevant Use Cases.



•Use Case diagram of uci.argo.**KERNEL**

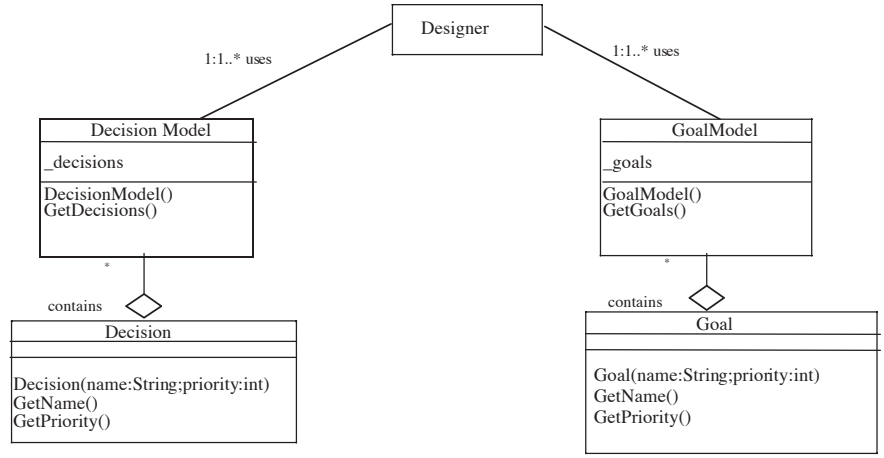
Step 2/1: *Analysis of Sequence and Class diagrams involved in the selected Use Case.*



•Sequence diafram of uci.argo.Kernel.**USERMODEL**

Step 2/2:

Class diagram analysis



Class diagram of uci.argo.Kernel.**USERMODEL**

Step 3: *Test Units definition*

Test Units: those system units separately testable for exercising a possible use of system:

Test Units:

DecisionModel, Decision, GoalModel, Goal.

Step 4: *Research of Settings and Interactions Categories*

- *SettingsCategories*: parameters and inputs relevant in the Test Unit.
- *InteractionsCategories*: objects interactions, exchanged messages.

Categories:

DecisionModel
Settings: _decisions
Interactions: DecisionModel()
getDecisions()
Decision
Interactions: Decision(name:String,priority:int)
GetName()
GetPriority()
.....

Step 5: *Test Specification construction*

- For each identified category, we find all its possible values and constraints.

Test Specification:

DecisionModel:

Settings: **_decisions**
Naming
Storage
Modularity
Inheritance
Stereotypes
Relationship.....

Interactions: **getDecisions()**
Opening a new file
Opening a saved file
After a modification and before saving
After a modification and after saving
DecisionModel()
Class constructor

Step 6: *Search of Message Sequences and Test Case definition*

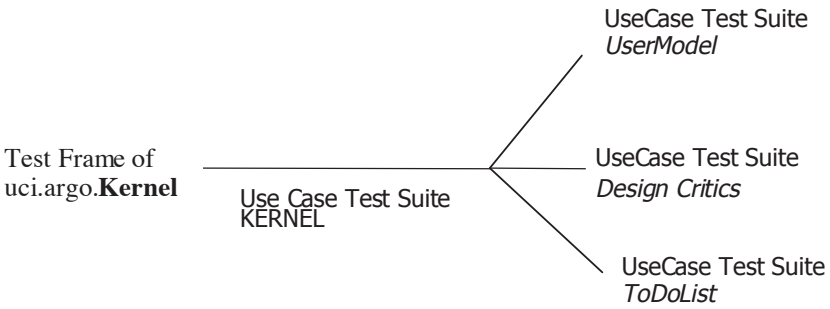
- **MessagesSequences:** set of Messages, involved in a Sequence diagram, used by objects to define and elaborate specific functionalities
- **Test Case:** is constructed, from a Test Specification, for each possible combination of choices of every category involved in a MessagesSequence

Test Case

getDecisions()
getName()/getPriority()
Opening a saved file
_decisions
Naming

Note: Such a Test Case must be repeated for all possible values of _decisions.

Step 7: *Definition of UseCase Test Suite and Incremental Construction of TestFrame*



Conclusions

- *Use Interaction method:* development of a systematic method and a tool for UML design-based integration test of complex system.
- Based exclusively on the use of UML diagrams:
  + immediately usable from industries already using UML (no additional modeling or design costs);
  + analysis can be started soon, and done contemporary with project development;
  - different interpretations of diagrams
- UI methodology itself designed using UML notation

Future work:

Validation on industrial real-world case study

Construction of a tool for automatizing part of this process and helping the tester in the construction of Test Case and Test Frame

Test tool integrated with the front-end design tool

Integrating this informal method with SA-based, formal methods, currently under investigation

A Practical approach to UML-based derivation of integration tests

F. Basanieri, A. Bertolino
IEI-CNR, Pisa, Italy

Abstract: *We present an on-going project for developing a tool supported test methodology based on UML descriptions. The leading criteria of this investigation are that no additional language or expertise is required more than UML, and that the methodology can be easily transferred to industrial contexts.*

Keywords:
Category-partition, Integration test strategy, UML, Use-Interaction test.

1. INTRODUCTION

In recent years, the object oriented (OO) paradigm has got widespread use. Software developers need appropriate OO models and support tools for describing and analysing the characteristics of complex distributed systems. OO system models usually involve graphical notations, so that the relationships between the many elements (objects) forming the system can be easily visualized.

UML, the Unified Modeling Language, is the emerging graphical notation to model, document and specify OO systems along all the phases of the software process. Indeed, there exist now many studies about using UML for design, and many tools are available. However, only a little part of these studies so far has addressed the usage of UML for testing, and none of the available commercial tools provide specific assistance for test planning and generation from the UML descriptions.

In this paper we address UML-based testing. Specifically, we want to address this task according to the following two guidelines: (i) we aim at a test method that is entirely based on UML, so that it can be easily adopted by industries already using UML; and, (ii) we refer to high level descriptions of the system: indeed, we want to address test planning for the integration test phase starting from the very first stages of system design.

We present here our practical approach for UML-based integration testing, that is called ***Use-Interaction testing***, as it mainly uses the UML Use Case and Interaction diagrams (specifically, the Sequence diagram). Indeed, integration testing is aimed at verifying that the (hopefully pre-tested) system components interact correctly, and UML Interaction diagrams can provide the information of how the system components should interact.

The proposed approach is very simple, and is inspired at large by the well-known Category Partition method [OB88]: we first look at the Use Case diagram to identify the suitable steps of an incremental test strategy. For each identified sub-Use Case, we then look at the Sequence diagram to identify the relevant components, or "Test Units". For each Test Unit, we derive the relevant "Settings" (could be parameters, variables, environmental states) and the relevant "Interactions" (essentially, messages from other Test Units), and identify for them the significant "Choices" (with the same meaning of the Category Partition method). To do this, we can analyse the related Class Diagrams, as well as other design documentation. Hence, by following the sequences of messages between the components over the Sequence Diagram, we construct the Test Cases, whereby each Test Case is characterized by a combination of all suitable choices of the involved Settings and Interactions.

We have applied the method to a case study, Argo/UML [Argo]. Argo/UML is an open source tool that provide cognitive support to system design. We have developed quite a few Test Cases, and we have been able to identify a few bugs in the case study. However, the method is currently manual, and still under evaluation. Our short term plan is the realization of a tool that support it, by smoothly expanding an existing UML design tool.

The paper is organized in the following way. In the next section, we provide some introductory material to UML and Integration testing. In

Section 3, we describe the method in more detail, explaining it step by step. Then, in Section 4 the case study Argo/UML is briefly introduced, so that in Section 5 some examples of application of Use-Interaction to it are shown. In Section 6 we also provide a very brief sketch of a UML definition of the Use Interaction method itself by means of the UML extension mechanism of stereotypes.

2. BACKGROUND

UML is a graphical modeling language to visualize, specify, design and document all the phases of a software development process. Born by the unification of I. Jacobson, J. Rumbaugh and G. Booch methods, in 1997 it was declared by OMT the standard for analysis and design of Object-Oriented systems.

2.1 UML diagrams

A UML design consists of an integrated meta-model composed of many elements representing OO common world concepts. Through this meta-model we define *Views* showing different aspects of the system to be modelled (Logical View, Component View, Deployment View, Concurrency View and Use Case View). As a whole, these views provide a complete picture of the system to be built. UML diagrams are graphs, each describing the content of a view; they can be arranged in different combinations to provide several system's views.

We only provide in this section a brief description of the UML diagrams mainly used in our methodology (for more details see [UML97a], [UML97b]):

Use Case diagram: a use case is the representation of a functionality (a specific use) provided by the system. This diagram shows a number of external users (actors) and their relationships with the system, when this is used to satisfy a specific functional requirement.

Class diagram: it shows the static structure of the system through the representation of its classes with their attributes and methods. Moreover it specifies the relations between the classes using different types of associations. A system can have more than one Class diagram; in subsequent phases of the software development process, the Class diagram represents objects at different levels of abstraction.

Sequence diagram: it shows the dynamic collaborations between a certain number of objects, highlighting the way in which a particular scenario[1] is realized using the interactions of a (sub)set of these objects. More precisely, a scenario is described by the set of messages exchanged between objects. A Sequence diagram expresses the same information of a Collaboration diagram, whereby the former describes the interactions between the objects during their life-cycle, while the latter shows the relative distribution of these objects links in the space.

2.2 Integration Testing and UML-based Testing

The testing goal is to execute the system to verify its behaviour and to reveal possible failures. Testing is an important piece of the software development process, because of its cost and impact on the reliability of final product. In our methodology we consider the Integration Testing phase, performed to find errors in unit interfaces and to build up the whole structure of software system in a systematic way.

To do this, one could use a non-incremental approach (big-bang), where all the modules are linked together and tested all at once, or preferably, incremental approaches like *top-down*, where modules are integrated from the main program downto the subordinated ones, or, *bottom-up*, where tests are constructed from modules at the lowest hierarchical level and then are linked together upwards, to construct the whole system.

But, when we consider an object-oriented system, the described techniques are not always usable, because, for example, we cannot identify the hierarchical structure of control by which it is possible to define top-down or bottom-up strategies. In an object-oriented environment we can test the class interactions by, for example, integrating together those classes used in reply to a particular input or system event (thread-based testing) or by testing together those classes that contribute to a particular use of the system. In the proposed methodology, the classes to be integration tested (modules, subsystem, processes) are those that realize a system functionality identified from a Use Case diagram.

[1] A scenario is defined as a specific sequence of actions that illustrates behaviour and may be used to show an interaction. [UML97b]

As a matter of fact, even though UML is a powerful mechanism of description, we have found few studies about its use to guide the testing phases.

Some researchers have proposed methods to translate a UML description into another formal description, and then derive the tests from the latter. For instance, in [JGP98] the authors present a tool, UMLAUT, that is used to manipulate the UML representation of the system and automatically transforms it into an intermediate form, suitable for validation. Another interesting paper is [OA99], where a method is proposed to generate test data from UML State diagrams. They translate the UML State diagram into formal SRC specifications, from which input data for unit testing are automatically generated. Finally, in a recent paper from Siemens Corporate Research [HIM00] UML diagrams are used to automatically construct test cases as follows. The developers first define the dynamic behaviour of each system component using a State diagram; the interactions between components are specified by annotating the State diagrams, and then the global FSM that corresponds to the integrated system behavior is used to generate the tests. This approach is being automated to execute the tests in an environment compatible with the UML modelling tool Rational Rose.

All the mentioned studies are interesting, and we see them as complementary to ours, as we do not use State diagrams, but system descriptions at coarser granularity.

3. AN INTEGRATION TEST APPROACH

3.1 Overview of the Use Interaction Testing approach.

As said, Integration Testing progressively verifies the interactions between software components (modules, packages, subsystem, processes, classes) in order to realize the final integrated system.

The ***Use Interaction*** methodology for Integration testing uses as a reference model the UML diagrams to systematically construct and define tests.

The *Use Case diagrams*, by visualizing the various system functionalities, help the tester to decide the way in which the system can be decomposed for testing each of its parts (representing a specific functionality of a Use Case) and then the whole system. Each Use Case can contain in turn other Use Cases, since, to obtain a complete system functionality, it is generally necessary to execute several actions realizing lower level functionalities. In our methodology, Use Case diagrams drive Integration test according to an incremental strategy. We start analysing low-level functionalities that represent a subset of actions described by a sub-Use Case, and then we progressively put them together, until the whole system described in the main Use Case is obtained.

For each selected Use Case we analyse the corresponding *Sequence diagram*, composed by objects and the messages they exchange. The objects involved in the diagram are those that realize and execute the functionality described in the Use Case through elaborations and message exchanges and so they are precisely the components to be tested. In this phase we consider the *Class diagram* too, and particularly a Class diagram at high level of abstraction. This diagram becomes important to define operations (or abstract operations) and attributes required by classes for the interactions of their objects. Also the Collaboration diagram could be used to analyse the object interactions, but it is not as expressive, since it underlines links and dependencies among objects, but not their dynamic interactions along a temporal sequence. Therefore, for our methodology, the most important diagram is Sequence diagram, that is the basis for generating integration tests. In [JBR98], in fact, we can find a suggestion to use this diagram in integration testing phase. They suggest to study different sequences found in this diagram from a possible input state, or from a system input done by actors.

To analyse the message sequences we have used a systematic methodology inspired by the well-known Category Partition Method [OB88].

3.2. The Category Partition method

Category Partition (CP) [OB88] is a well-known method to systematically derive functional tests from the specifications.

Generally speaking, the partitioning of the input domain is a standard approach to functional testing, based on the idea that, for the classes of equivalence defined by the identified partitions, one or few tests can be selected as representative of the whole class behaviour.

The first step of the CP method is to analyse the functional requirements to divide the analysed system in *functional units* to be separately tested. A functional unit can be a high-level function or a procedure of the implemented system. For each defined functional unit, the *environment conditions* (system characteristic of a certain functional unit) and the *parameters* (explicit input of the same unit) relevant for testing must be identified. *Test Cases* are then derived by finding significant values of environment conditions and parameters; this can be done dividing them into *categories* representing relevant system properties or particular characteristics of parameters or environment conditions. Then, for each category, we identify different *choices*, that are different significant values for these categories. The CP method uses a tool to automatically construct *test cases* from specifications expressed into a dedicated semi-formal specification language, called TSL. The CP method has encountered wide interest, and has inspired the development of a large number of test methodologies, also using formal languages such as Z.

3.3 Steps for Use-Interaction testing

We now describe in more detail the proposed approach. It can be logically subdivided into seven steps.

Step 1: *UML Design analysis and search of different Use Cases.* We analyse UML design and, particularly, Use Case diagram. This diagram can lead test construction by defining different integration test stages with respect to an incremental strategy from lower-level abstraction levels upto higher ones. Inside this diagram we can thus find different Use Cases representing (sub)functionalities used to construct the main one (see, for example, Fig.1)

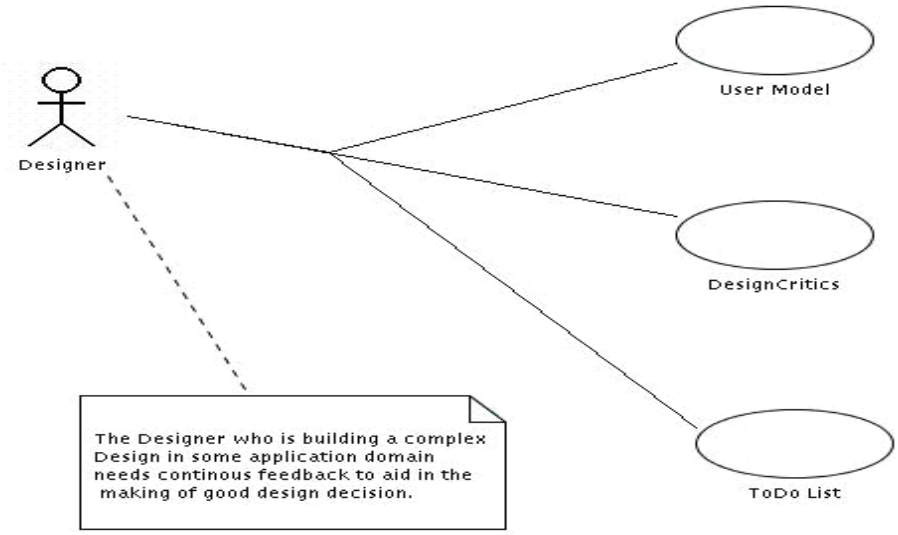


Fig 1: Use Case diagram (of uci.argo.Kernel)

Step 2: *Analysis of Sequence and Class diagrams involved in the selected Use Case.* After selecting one of the Use Cases, we analyse the relative Sequence (Fig. 2) and Class (Fig. 3) (at a high level of abstraction) diagrams. The focus is mainly on the Sequence diagram, analysed along two orthogonal directions corresponding to its axes. The horizontal axis (Arrow a) shows a set of objects that interact through messages; this axis is used to verify the object interactions and their correct use with respect to the Use Case requirements. The vertical axis (Arrow b) is studied because it shows the temporal sequence, from top to bottom along the objects life time, of messages involved in object interactions.

Moreover, horizontal axis is useful for tracing the message sequences, that is at the basis of test cases construction.

Step 3: *Test Units definition.* Each object inside a Sequence diagram is considered a *Test Unit*, since it can be separately tested and it represents and defines a possible use of system. Test Units search can be done automatically because it descends directly from a Sequence diagram.

Step 4: *Research of Settings and Interactions Categories. Interactions categories* are the

interactions that an object has with the others involved in a same Sequence diagram and they are represented by all the messages to the considered object. In this type of diagram, in fact, the interactions are defined with a sender who sends a message to a receiver asking for a service provided by the latter. So, following vertically the life time of the analysed object, we can find all its Interaction Categories represented by the entering arrows. We could also study these methods in the Class diagram description, but it is for searching the *Settings Categories* that the Class diagram becomes very important. Settings categories are attributes (or a subset of them) of a class (and the corresponding Sequence diagram's object), like input parameters used in messages or data structures.

Step 5: *Test Specification construction.* In this step we define a *Test Specification* involved in a Test Unit, that is: for each found category we find all its possible values and constraints. For this aim we use the Class diagram where we can find a preliminary description of a method implementation, its possible input values or the description of an attribute used and its significant values.

Step 6: *Search of Messages Sequences and Test Cases definition.* Following the temporal order of the messages involved in a Sequence diagram, it is possible to find some *Messages Sequences,* i.e., a set of messages used by objects to define and elaborate particular functionalities. Several categories can correspond to each found Message Sequence, of the two types Interaction and Settings: precisely, messages/methods in the considered sequence identify the Interaction Categories, and the attributes that affect these messages identify the Settings Categories. For each category (of either types) within a Message Sequence, we consider each possible choice, taking them from the Test Specification (Step 5). Then, we derive as many *Test Cases* as necessary, by considering all potential combinations of compatible choices. The construction of Test Cases could be performed by a tool, after Messages Sequences and Test Specification have been appropriately defined.

Step 7: *Definition of Use Case Test Suite and Incremental Construction of Test Frame*. Finally, all Test Cases built for a same Use Case are collected together into a *Use Case Test Suite*. If there exist more than one Use Cases to be analysed at the same level of abstraction, we repeat the process from step 2, constructing other Use Case Test Suites.

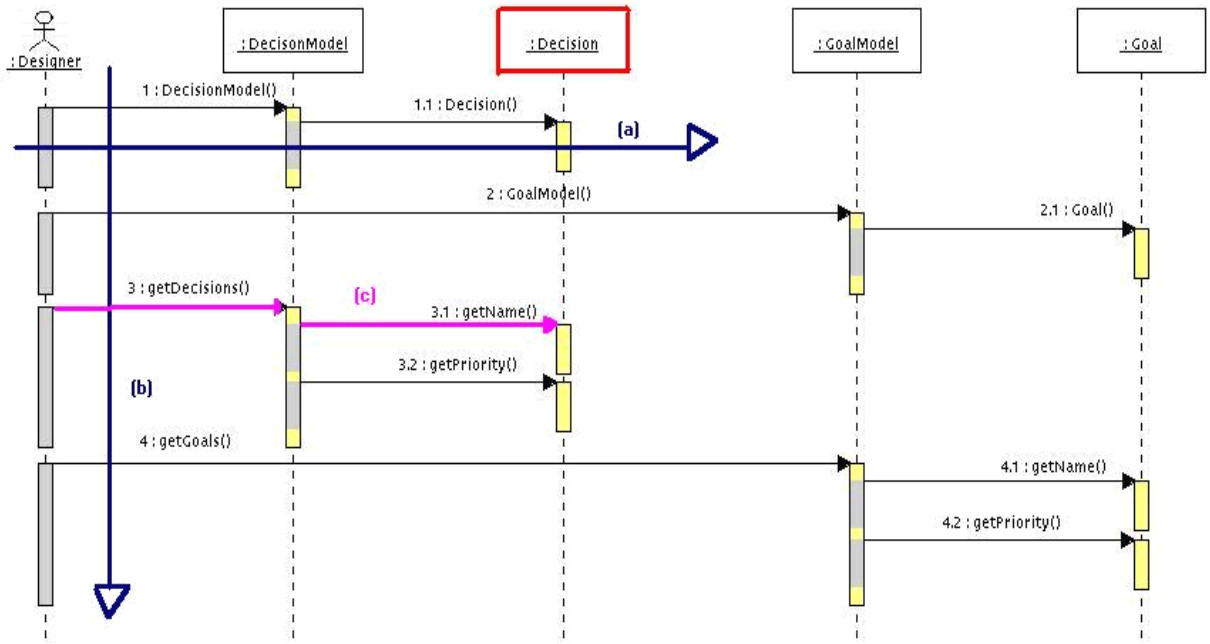


Fig 2: Sequence diagram analysis

After all the (sub)-Use Cases, at a given level, have been analysed, we consider other possible Use Cases at higher levels of abstraction, based on the Use Case diagrams, until we reach the main Use

Case. The same seven-steps process is applied at the identified (higher) Use Cases, using again the Sequence and Class diagrams. As we move towards the main Use Case, the Test Units in the Sequence diagram will generally be more abstract system components than those considered for the (sub)-Use Cases.

Finally, we define a *Test Frame*, that is, the set of all Use Case Test Suites built for the analysed (sub)system.

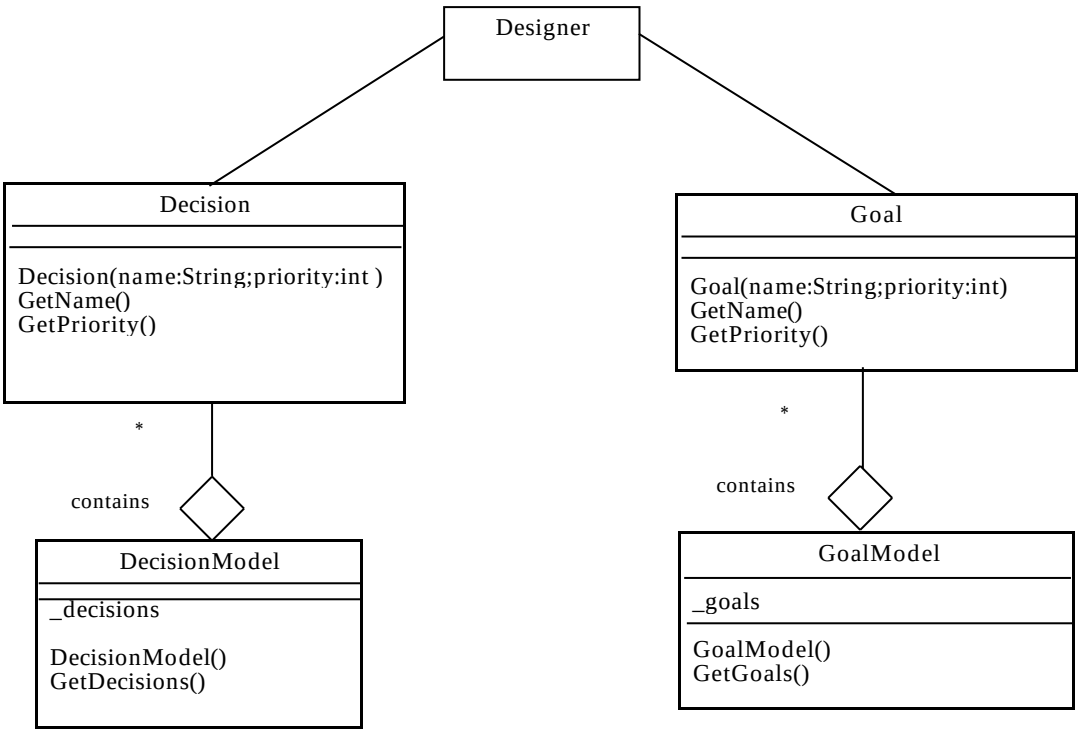


Fig 3: Class diagram of the selected Use case for uci.argo.UserModel

4. THE CASE STUDY

The analysed case study is Argo/UML [Argo], a tool supporting Object Oriented design with UML; this is an open source project that was launched by a research group of the Institute of Computer Science of Irvine, University of California. Argo/UML is based on the UML 1.1 specification and uses a Java version of UML meta-model with support for OCL and XMI (XML Model Interchange format).

We selected this case study because it follows exactly the UML specifications, and it is completely UML designed with good documentation on Java code, even if only a part of its UML design is available on the web. This tool is designed to provide very interesting features, but most of them are not yet developed and only three diagrams are supported so far: Use Case, Class and State Machine diagrams. Its major feature is to provide "cognitive support for design", that is an intelligent support for designers who have to build a complex software product. This support, as the authors describe, is based on: (a) Reflection-in-action; (b) Opportunistic Design.

(a)Reflection-in-action is divided into: Design Critics: simple agents that continuously execute design analysis in a background thread of control, while the designer is working, and suggest possible improvements. The suggestions can be, for instance, the signalling of syntax errors or reminders to return to parts of the design that needed to be finishing; Corrective automations: done by a Wizard for critics identifying specific problems in the design; To Do List: items, organized in a To Do List, helping designer about many details of his work. They are, for example, suggestions from arisen critics, personal notes, suggestion for improving and completing some parts and so on; User Model: maintains information about designer's choices about project and its features. This is done, for example, suggesting only critics that are relevant for designer's tasks. User model consists on a **decision model**, a list of decisions that must be made when doing object-oriented design and **goal model**, a list of goals that designer must reach for the design project. (b) Opportunistic Design divided into: To Do List: a list of To Do Item; CheckList: used to remind designers to cover all design details and avoid

common errors. These lists are specific to the selected design element (association, classes, attributes..).

For our case study we have chosen the part of the tool realizing cognitive support for design, the *Argo* package, divided into two other packages: *uci.argo.Kernel*, that manages Critics, User Model and To Do List and *uci.argo.Checklist*, that manages Checklist. In this paper, for reason of space, we analyse only uci.argo.Kernel. Its functionalities are divided into: **Design Critics**: that manages critics, their definition and use during design. Usable critics are visible in Critic Browser, Fig.4, that keeps track of all information like critic state (active or inactive), priority, general description and so on. All critics properties can be changed by users but it implies changes in To Do items, where you can keep track, every time, of active critics and their priority values. **To Do List**: manages To Do Items, Fig. 5; it shows all active critics descriptions and steps to eliminate them. **User Model**: manages decisions (Decision Model) and goals (Goal Model) made by users for project. **Decision Model**, Fig. 6, is used to specify which requirements are important for designer's aim; they have a priority value from 0 to 5 and if a decision priority is 0 the corresponding critics are made inactive. **Goal Model** shows which goals designer would achieve, Fig. 6. Now only a goal type (Unspecified) is usable; like decisions, goals have a priority from 0 to 5 and, if a goal has priority 0 the related critic is made inactive.

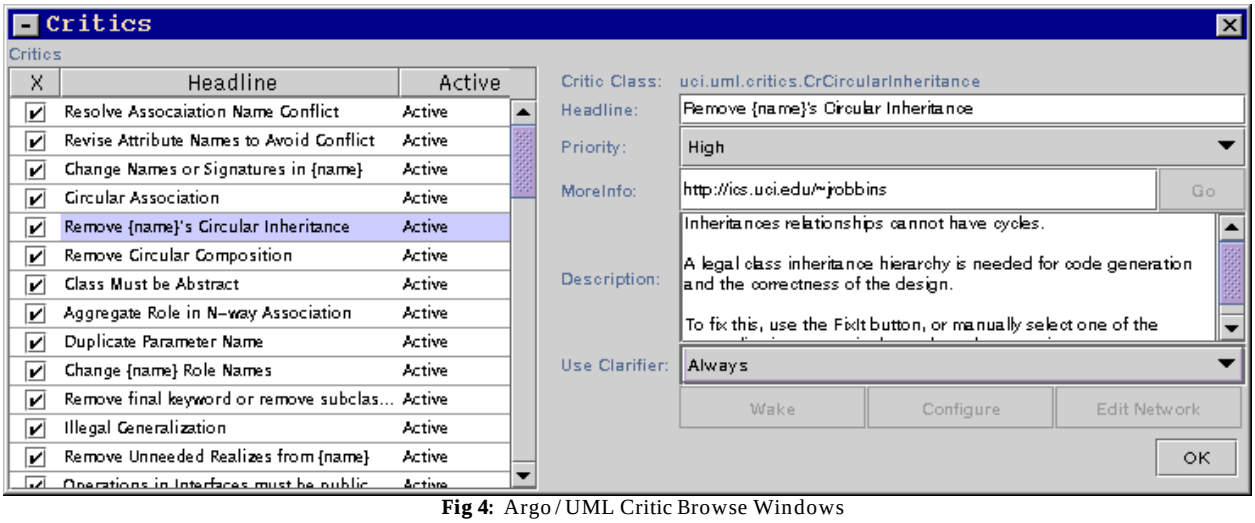


Fig 4: Argo / UML Critic Browse Windows

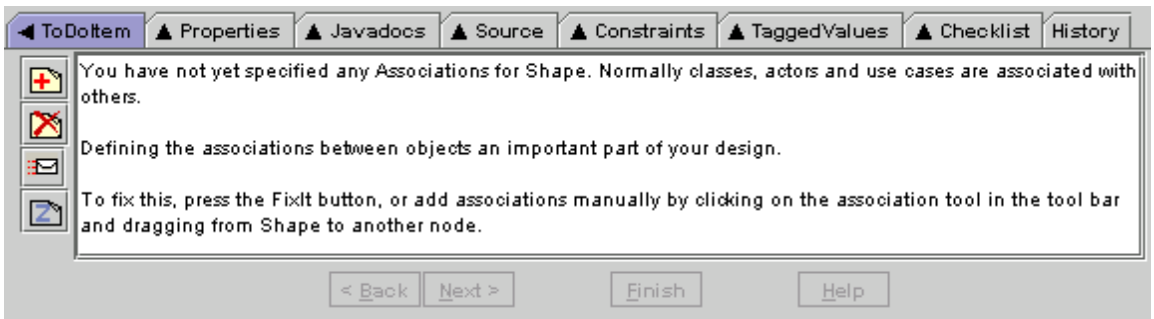


Fig 5: Argo / UML To Do Pane

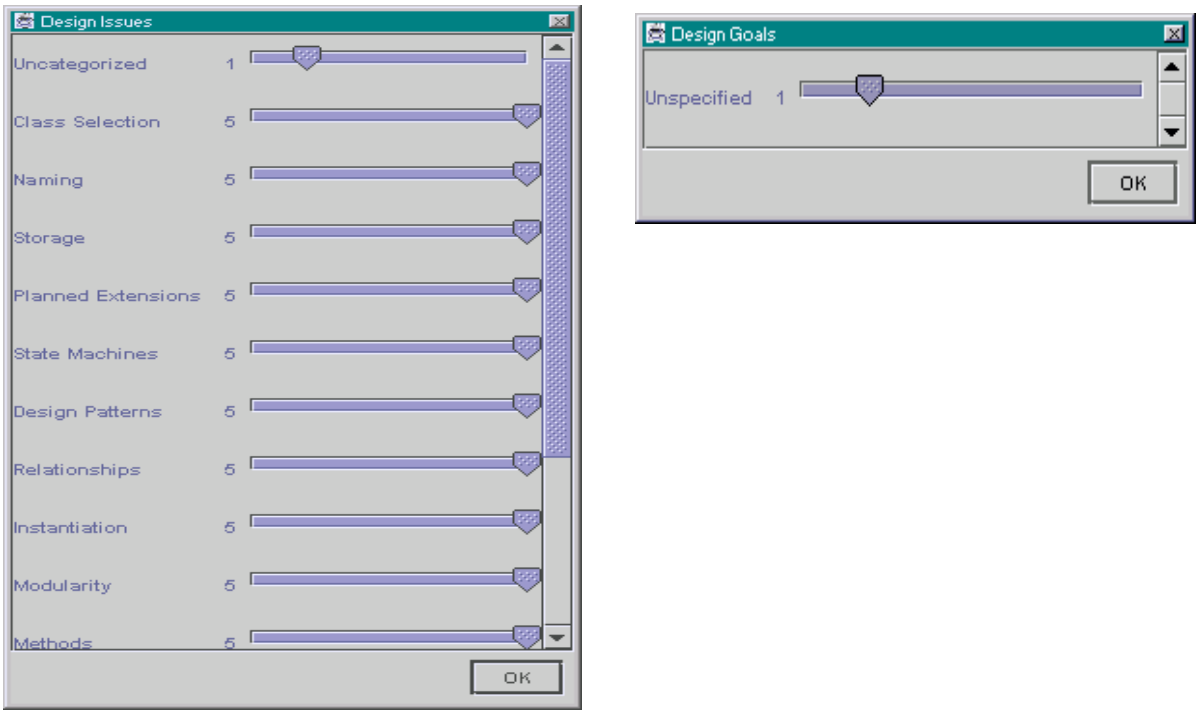


Fig 6: Argo / UML Design Issues Panel and Design Goals Panel

5. APPLICATION OF USE INTERACTION METHODOLOGY TO THE ARGO CASE STUDY

Step 1: *UML Design analysis and search of different Use Cases.* The Use Case diagram of uci.argo.Kernel is shown in Fig. 1. We can see that the main functionality of Kernel is divided into three sub-functionalities, each of which can represent a single sub-Use Case. These Use Cases can be analysed separately to construct the Test Units and Test Cases of a Use Case Test Suite, and then linked together to build one main Test Frame. In this section we analyse only the User Model functionality.

Step 2: *Analysis of Sequence and Class diagrams involved in the selected Use Case.* Fig. 2 and 3 show Sequence and Class diagram for uci.argo.Kernel.UserModel.

Step 3*: Test Units definition.*

TestUnits:

DecisionModel, Decision, GoalModel, Goal.

Step 4: *Research of Settings and Interactions Categories.*

Categories**:**

DecisionModel

setting: _decisions

interactions: DecisionModel(), getDecisions()

Decision

interactions: Decision(name:String,priority:int) ,
GetName(), GetPriority()

GoalModel

settings:_goals

*interactions:*GoalModel(), getGoals()

Goal

interactions: Goal(name:String, priority:int),
GetName(),GetPriority()

Step 5: *Test Specification construction.* We describe the Decision Model Test Unit.

Decision Model: is a part of Design state, it describes which type of decisions are useful for designer; critics relevant for these decisions become active.

Settings: _decision
Naming, Storage, Stereotypes, Inheritance
Relationship, Modularity, ….

Interactions:

GetDecisions()
Opening a new file
Opening a saved file
After a modification and before saving
After a modification and after saving

DecisionModel()
Constructor of class.

Step 6: *Search of Messages Sequences and Test Cases definition.* One of the possible Messages Sequences, highlighted in Fig.2 by Arrow c, is:
getDecisions() -> getName / getPriority(),
and the corresponding Test Case for a possible choice involved categories is:

TEST CASE

getDecisions()

getName()/getPriority()

Opening a saved file.

_decisions = Naming

Action to perform test:Visualizing Design Issues Panel for the considering decision and priority opening a saved document.

Instructions for checking the test: opening the Design Issues Panel we see decision with Name=Naming and priority like priorità of the same file previously saved.

Note 1: This Test must be repeated for all possibile values of _decisions.

Step 7: *Definition of Use Case Test Suite and Incremental Construction of Test Frame.* After defining Test Cases we build a Use Case Test Suite for User Model, Design Critics, To Do List and then, following an incremental integration test strategy, we analyse the main functionality of Kernel constructing its Use Case Test Suite and, at the end, the entire Test Frame.

Results from the case study: when the method has been applied to case study Argo/UML some bugs were discovered. We show two Test Results: the first one, without bugs, is referred to previous Test Case (Step 6); the second is an example of a test with fail result.

TEST CASE 1

getDecisions()

getName()/getPriority()

Opening a saved document.

_decisions = Naming

Test Result: Passed.

TEST CASE 2

SetDecisionPriority (priority:int)

From a >0 value to 0 before saving.

BeInactive()

Critic is active.

RemoveItems(item:ToDoItem)

Critic is disactivated.

RecomputeAllToDoITems()

A decision has priority 0.

_decisions: Naming.

Test Result: Failed.

Fail report: Test is made for all **_decision** values**.** For each of these related critics are disactivated but one critic is never disactived. This is critic 28: "Add operation to <ocl>self</ocl>". This critic is linked to decision *Behavior*, but this type of decision does not exist among built decisions and inside DesignIssuesPanel. Moreover, in the related ToDoPane the critic, even if active, is not visible since it is not linked to any existing decision. The critic can be disactivated only manually from Critic Browse Window.

6. UML DEFINITION OF METHODOLOGY

The Use Interaction testing methodology follows a precise process, and therefore can itself be designed using the UML language. We show, in Fig. 7, a Use Case diagram in which the actor is Tester who interacts with system generating test cases.

Moreover, all the new introduced concepts, like Test Unit or Setting and Interactions Categories, have been defined using exthension mechanism of stereotypes. A stereotype [UML97a] is a new type of modeling element that extends the

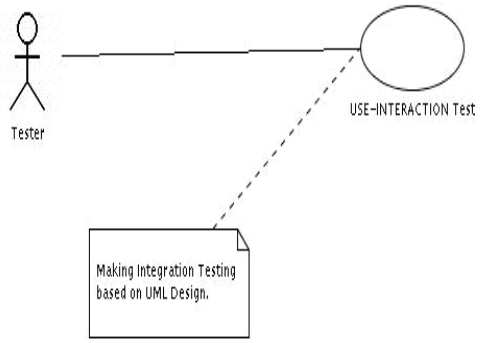
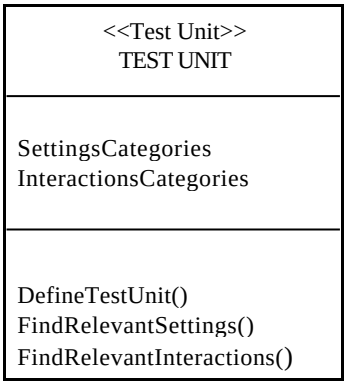


Fig. 7: Use Case diagram of *Use Interaction Test* method.

semantic of the existing metamodel; all our new concepts are (sub)classes of the existing metaclass Class with new additional characteristics and constraints. For instance, we show a stereotyped class, Test Unit, with its properties and constraints defined using the OCL language [UML97c].

Test Unit:



Stereotype Test Unit for instances of meta-class Class

[1] For each Test Unit there exists one and only one related Classifier Role.

self_forAll(t: TestUnit | self.has_exists(c1: ClassifierRole | c1.name = t.name and forAll(c2| c1<>c2 implies c2.name <>t.name)))

[2] The attributes of a TestUnit are SettingsCategories and InteractionsCategories.

self.Attribute.ocltype = enum{SettingsCategories, InteractionsCategories}

[3] A SettingsCategory represents one of the class attributes and it is involved in one iteration.

self.SettingsCategory_exists(a Attribute| a.isused=self and a IsInvolvedIn Interactions)

[4] InteractionsCategories are the only associations of CollaborationDiagram.

self.InteractionsCategory_forAll(i|self.ocltype.Collaboration.Association_exists(ass| ass.name = i))

7. CONCLUSIONS

In this paper we have presented our methodology, *Use Interaction Test*, that generates Integration Tests from UML diagrams like Use Case and Interactions diagrams.

This testing approach is placed inside a larger research project trying to develop methods and tools for design-based integration test of complex system. The approach used in this work is not based on formal methods for specification, like the related [BCIM00], [MLB99] works, but only on exclusive use of UML diagrams.

For this reason, we think it may be a viable method for its simplicity and easy portability to industrial contexts; moreover, since we use only UML diagrams, the methods does not require specialised expertise and analysis, and so generation of test cases can be done contemporary with project development, at no or little extra cost.

On the other hand, the exclusive use of UML diagrams can be considered also a limit of our method, because as known UML semantic is not very precise, and therefore the diagrams can have different interpretations from different users, and also different designers could provide different diagram specifications.

In future, we aim at constructing a tool automatizing part of this process in collaboration with industrial partners. The tool would help the Tester in Test Cases generation for the tasks not requiring human judgement (for example, identification of Test Unit and Messages Sequences), thus enhancing the cost effectiveness of the approach.

Acknowledgements

This work is supported in part by the Italian Murst Project "**Saladin**: *Software Architecture and Languages to coordinate Distributed Mobile Components"*.

References:

[Argo] "Argo/UML", available from http://argouml.tigris.org.

[BCIM00] A. Bertolino, F. Corradini, P. Inverardi and H. Muccini, "Deriving Test Plans from Architectural Descriptions"", *Proc. ACM/IEEE 22nd Int. Conf. on Soft. Eng. (ICSE 2000),* Limerick, 4-11 June 2000, pp 220-229.

[HIM00] J. Hartmann, C. Imoberdof, M. Meisenger, "UML-Based Integration Testing", Proceedings of ISSTA 2000, Portland, Oregon, 22-25 August 2000.

[JBR98] I. Jacobson, G. Booch, J.Rumbaugh, *The Unified Software Development Process,* Addison-Wesley, 1998.

[JGP98] J.M Jézéquel, A. Le Guennec, F. Pennanech, "Validating Distributed Software Modeled with UML", Proc. UML98, in *LNCS 1618,* pp. 365-376.

[MLB99] F. Mercier, P. Le Gall, A. Bertolino, Formalizing integration test strategies for distributed systems", *First Int. ICSE Workshop Testing Distributed Component-Based System,* Los Angeles, May 1999.

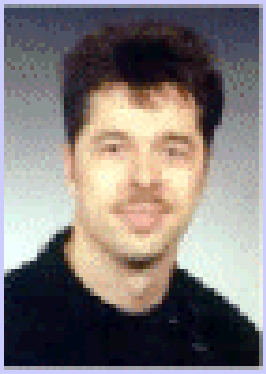
[OA99] J. Offutt, A. Abdurazik, "Generating Test from UML Specifications", Second International Conference on the Unified Modeling Language, UML 99, Fort Collins, CO, October 1999.

[OB88] T.J. Ostrand, M.J. Balcer, "The Category Partition Method For Specifying and Generating Functional Tests", Communication of the ACM, 31(6), p. 676-686, June 1988.

[UML97a] UML Notation Guide, v. 1.1, Sept. 1997, *www.rational.com/uml/resources/documentation/formats.jtmpl*

[UML97b] UML Semantics, v. 1.1, Sept. 1997, *www.rational.com/uml/resources/documentation/formats.jtmpl*

[UML97c] Object Constraint Language Specification, version 1.1, Sept. 1997, *www.rational.com/uml/resources/documentation/formats.jtmpl*



QWE2000 Session 3A

Mr. Rob Hendriks & Mr. Robert vanVonderen & Mr. Erik van Veenendaal [Netherlands] (Improve Quality Services)

"Measuring Software Product Quality During Testing"

Key Points

- ISO-9126
- Quality requirements
- Quality characteristics
- Testing
- Practical experience

Presentation Abstract

Quality requirements of software products are often described in vague and broad terms. As a consequence it makes it difficult for software engineers to determine how quality influences their assignment and is it almost impossible for test engineers to evaluate the quality of the software product as no concrete and quantitative reference, of what quality in that context means, exists. The International Organization for Standardization (ISO) has defined a set of quality characteristics to enable the definition of software product quality in terms of functionality, reliability, usability, efficiency, maintainability and portability. These quality characteristics are described and defined in the ISO-9126 standard. In various annexes to this standard metrics are provided to actually measure the characteristics. Within Océ Technologies B.V., a Dutch developer and manufacturer of copying and printing equipment, the ISO-9126 standard has been used in the testing phase, to specify the software quality targets for the newest line of copier controller software. This copier controller software is a multi-site development project, which takes place in the Netherlands and France. Approximately 60 software engineers are involved in the project. This paper depicts, step-by-step, the actions taken to implement ISO-9126 in the project organisation and the results that were obtained. Described is how the most important quality characteristics (functionality, reliability and maintainability) were selected, by means of a questionnaire developed for the European SPACE-UFO project, to form a quality model. The next step on how to define and select the metrics to measure on the quality characteristics is described, including the determination of a baseline value for each metric. Finally the actual measurement during the test execution and the evaluation of the results obtained are discussed. During the evaluation of the results the relevance of the various metrics

and the baseline values were analyzed and, if necessary, modified. An evaluation of the advantages and disadvantages of the approach taken concludes the paper.

About the Speaker

Rob Hendriks has several years of experience within the area of software quality for embedded software systems, with a specialisation on software testing. The past years he has been working as a test co-ordinator and consultant in projects for consumer electronics and professional systems. On a regular basis he gives training in the area of software testing and has a specialisation in testing techniques. He is the co-author of the paper 'Structured testing of embedded software' published in Software Release Magazine in February 2000. Rob is co-founder of WEB, a study group for embedded software testing, which is a co-operation of companies, both development as service organisations, working in the area of technical automation.

Robert L.A.H. van Vonderen, MsC, has worked for Océ Technologies B.V. since his graduation from the Eindhoven University of Technology. He conducted a range of applied research studies in the area of embedded software, geographical information systems and printer controller software. In recent years he has worked as a project manager and project leader for the Océ Printing Systems division in the area of printer controllers and maintenance. He is currently responsible for the integration, testing and QA activities of the newest line of controller development within the Océ Document Printing Systems division.

Measuring software product quality during testing

Rob Hendriks[1]
Robert van Vonderen[2]

[1] Improve Quality Services B.V.
[2] Océ Technologies B.V.

Measuring software product quality during testing

Presentation outline

- Introduction/context
- Selection of product quality characteristics
- Definition of quality metrics
- Measurement
- Results and conclusion
- Questions

Introduction/Context

- **Objective: measure objectively software quality for copier controller**
  - Necessary to define software quality unambiguously
- **Applied at Océ Technologies copier controller software development**
  - Multi-site (NL/F)
  - 60 developers
  - new technology
  - incremental development (V-model)

3

November 22, 2000

IMPROVE Quality Services

Introduction/Context

- **Controller software test team**
  - **verify correct functional behavior**
  - **determine software product quality**
  - **but…**
- **No software quality requirements**

4

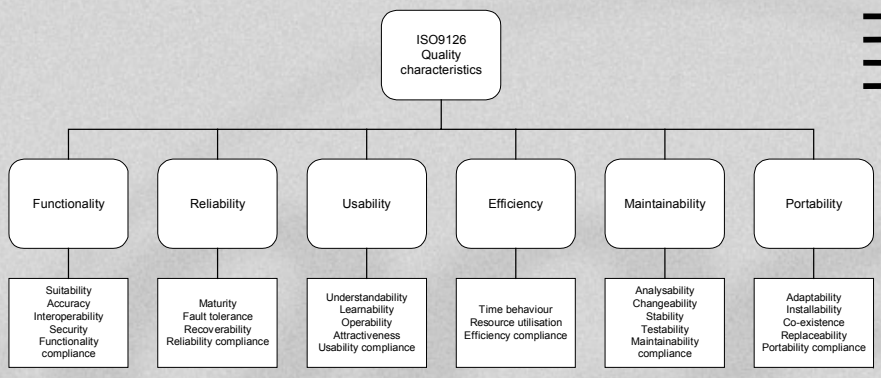
November 22, 2000

IMPROVE Quality Services

Selection of product quality characteristics

- **Determine software quality requirements**
  - **from the total product quality requirements**
    - MCBF, ACPR
  - **from ISO9126**
    - functionality
    - reliability
    - usability
    - efficiency
    - maintainability
    - portability

ISO9126



Selection of product quality characteristics

- **Not all quality characteristics equally important**
- ***Selection* by interviewing key-persons**
  - **project manager**
  - **software architect**
  - **types of users**
    - operator
    - service technician
    - system administrator
- **Use of a questionnaire**

7

November 22, 2000

IMPROVE Quality Services

Selection of product quality characteristics

Example from questionnaire:

(SPACE-UFO / Improve Quality Services B.V.)

- **Process related questions, e.g.:**

  What kind of market type is the product oriented towards? (Business/Consumer)

  (Related to suitability, interoperability, learnability, resource utilisation and time behaviour)

- **User related questions, e.g.:**

  What is the average age of the users?
  - Under 25
  - 25 – 40
  - Older than 40

  (Related to learnability and attractiveness)

8

November 22, 2000

IMPROVE Quality Services

Selection of product quality characteristics

Questionnaire results

	Score = 1	Score = 2	Score = 3	Score = 4	Score = 5
Functionality				X	
Reliability				X	
Usability			X		
Efficiency		X			
Maintainability			X		
Portability	X				

IMPROVE Quality Services

Definition of quality metrics

- Metrics per quality characteristic
- ISO-9126-2 and ISO-9126-3
- Extend with product-specific metrics
- In total 15 ISO metrics selected and 1 additional metric defined

IMPROVE Quality Services

Examples of metrics defined

Quality characteristic	Sub-characteristic	Metric	Purpose
Functionality	Suitability	*Functional implementation completeness:* Number of missing functions detected during system testing / Number of functions described in requirement specifications.	How many functions have been implemented in relation to the number of functions specified in the requirement specifications?
Reliability	Maturity	*Mean Copies Between Failures:* Total number of copies during system testing / Number of defects, caused by controller software, detected during operation time.	How frequent are the defects of the controller software in operation?
Maintainability	Analysability	*Availability of design documentation:* Available (and approved) design documentation (i.e. SW architecture, top-level design, analysis views, design views and interface specifications) / Identified design documentation.	What's the proportion of design documentation available?

Measurement

- Baseline for every metric
- Ask experienced people to estimate baseline
- 'Minimum but sufficient level'

Measurement

- **Measure during system test phase of each development cycle**
  - clearly instruct test engineers what data to record per defect, e.g.:
    number of copies made
    total down time of the system
- **Compare against baseline and previous cycle**

Results and conclusions

- **4 measurements done**
  - **cycles C1, C2, C2patch, C3**

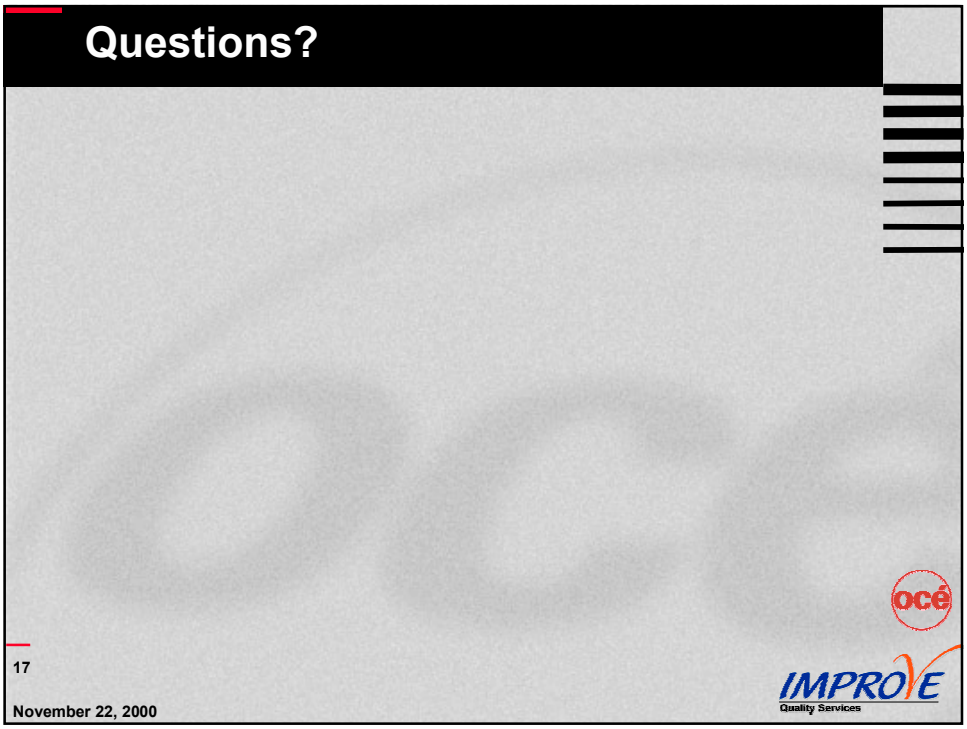
Quality-characteristic	Sub-characteristic	Metric	Base-line	Value C1	Value C2	Value C2 patch	Value C3
Functionality	Suitability	Functional implementation completeness	0.90	0.91	1.0	1.0	0.82
		Functional implementation correctness	0.80	0.45	0.80	0.84	0.61
Reliability	Maturity	Defect detection	0.75	0.46	0.63	0.63	1.08
		Mean copies between failures	100000	n.a.	93	175	4880
		Test completeness	0.90	0.75	0.78	0.33	0.92
	Recoverability	Mean down time	10 min.	n.a.	5 min.	5 min.	5 min.

Results and conclusions

- Not all metrics could be measured each cycle
- Perception of quality does not always show in the metrics (C2 vs. C2patch)
- Correlation between metrics and improvements / changes in development approach hard to determine
- Calculation of expected defect rate incorrect (C3 value higher than 1): to be investigated

Results and conclusions

- Relative short measurement period (less then 1 year) - with prolonged tests planned, our metric accuracy will improve
- Currently, no corrective actions are taken based on the measured results
- No real-life experience what happens with insufficient quality metrics at release date
- Lift experience with software quality measurement to total product quality measurement
- Continue!!!



Measuring software product quality during testing

Rob Hendriks, Robert van Vonderen and Erik van Veenendaal

Quality requirements of software products are often described in vague and broad terms. As a consequence it makes it difficult for software engineers to determine how quality influences their assignment and it is almost impossible for test engineers to evaluate the quality of the software product as no concrete and quantitative reference, of what quality in that context means, exists. This paper describes a possible way to define and measure software product quality. The authors have applied this method during the development of copier/printer controller software at Océ Technologies B.V., a Dutch developer and manufacturer of copying and printing equipment.

Context

In 1997, Océ Technologies started the development of a new line of copier/printer controllers, to be used in a new family of monochrome high-volume hybrid copier/printers. In April 1999 this development entered the engineering phase. The engineering of the software for this controller line takes place on three sites, of which two are in The Netherlands and one is in France. Approximately 60 software engineers are involved in the development of this software, hereafter referred to as the ‘controller software’. The controller software is developed in an incremental fashion, with each development cycle conforming to the V-model. Each iteration of such a development cycle takes between 3 and 5 months.

At the start of the engineering phase a test team was formed with the assignment to verify correct functional behaviour and to determine product quality of the controller software. One of the main problems that occurred for the test team was the fact that the required quality level of the controller software was not specified. The only quality requirements available referred to the copier/printer product as a whole and not particular to it’s software components. Those requirements were however easily measurable and easy to understand. Examples are the mean number of copies between failures (MCBF, which pertain to all system errors) and the average copies per repair (ACPR, which is the number of copies made between visits of a service technician).

The objective of the test team was to get a clear baseline of the quality requirements before the testing phase would actually start. Therefore a quality model had to be found that could help in defining and measuring software product quality. This model was found in ISO9126 (ISO/IEC 9126-1:2000). In this ISO standard six quality characteristics are defined, which help in decomposing quality in manageable parts. The quality

characteristics defined in the ISO9126 standard are functionality, reliability, usability, efficiency, maintainability and portability. To have a more detailed description these quality characteristics are divided into 27 sub-characteristics (Figure 1) in total.

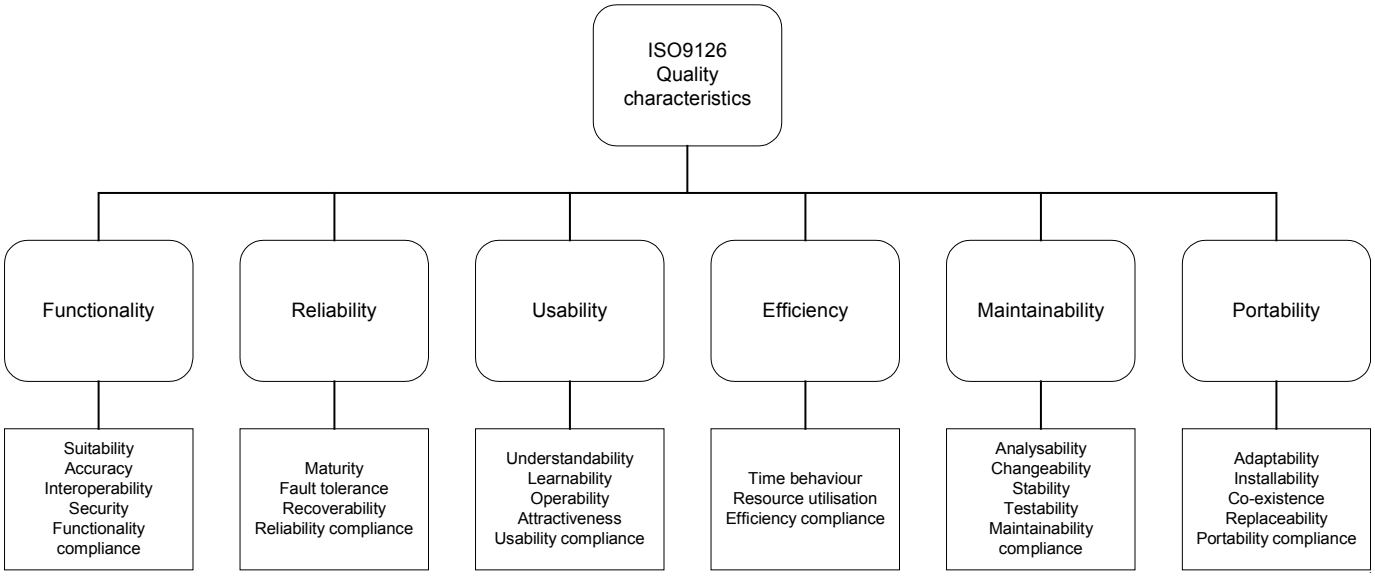


Figure 1 ISO9126 quality characteristics overview

Selection of product quality characteristics

Not all quality characteristics are of equal importance to the software product. Portability might be unimportant when the development aims at a dedicated platform and maintainability might not be an issue when it's a throwaway product. The important quality (sub-)characteristics need therefore be selected. This selection can be made by interviewing key persons in- and outside the project. Inside the project one can think of the product manager, the project manager or the software architect. Outside the project the various types of users are important. Copier users are not only limited to the person operating the copier, often forgotten are e.g. the service technician, the system administrator, etc.

The quality characteristics as defined by the ISO9126 standard are not always easy to interpret. What is meant by maintainability, or even worse usability? It's difficult to express these quality characteristics in an unambiguous way. As it will be hard to understand for IT professionals it will be even harder for users of the copier, who, in general, have no or limited knowledge of software. Most of the users don't even perceive that the product contains software. It will therefore be difficult to determine those quality characteristics that are important for the software component of the product just by asking "Do you think usability is important?" The persons interviewed have their own definition of the characteristics and

their own view of what the system should do and therefore what is important.

To overcome the problems mentioned above a questionnaire based method has been developed in the European SPACE-UFO project, funded by the European Union, and further elaborated by Improve Quality Services B.V.. This questionnaire consists of a number of questions about general product characteristics. These product characteristics are understandable for all interviewed persons. Instead of asking whether usability is important one asks questions about the characteristics of the users that influence the usability requirements, e.g. the number of different users, their experience with the product (or a similar one) and their educational level. Similar questions are asked for the other quality characteristics. The answers given are used to deduct the most relevant quality (sub-)characteristics. A fragment of the questionnaire is included in appendix A. This appendix also shows the related quality (sub-) characteristics.

Within the project 4 key-persons were selected and also 3 representatives of the different types of users were selected. Thus 7 persons were interviewed, each having their typical view on the copier/printer product. From the answers given a ranking for the product quality (sub-) characteristics could be deducted. A score from 1 to 5 was given, with 1 being unimportant and 5 most important. The results then were averaged for all respondents. This resulted in the score shown in table 1.

	Score = 1	Score = 2	Score = 3	Score = 4	Score = 5
Functionality				X	
Reliability				X	
Usability			X		
Efficiency		X			
Maintainability			X		
Portability	X				

Table 1 Score per quality characteristic

As can be seen from the results of the questionnaire, the quality characteristics functionality, reliability and maintainability were considered to be important for the controller software. This is explainable from the type of copier the project is developing: it is expected to run in a highly professional document production environment, where the uptime (to be expressed in ‘reliability’ and ‘maintainability’) is of utmost importance. Also, the copier/printer will be the successor product of an analogue high-volume copier model, where a clear functionality demand (being compatible with existing products as well as providing an extension) is expressed. Usability also scores high but it has been decided not to take it into account for testing the controller software. The reason for this is that

usability is considered to be a property of the user interface, which is not part of the controller software development.

Given this, the test team decided to focus on three quality characteristics (functionality, reliability and maintainability) primarily, and to develop quality metrics for these quality characteristics.

Definition of quality metrics

When the relevant quality (sub-)characteristics have been determined, one should think about how to measure quality. The ISO9126 standard defines, in the technical reports ISO 9126-2 (external metrics) and ISO 9126-3 (internal metrics), a number of metrics per quality characteristic that can be used for measurement.

A selection of metrics can be made and, when necessary, extended with self-defined metrics. The latter only on condition that the metrics are motivated by defining the goal and the attribute that will be measured.

For each quality sub-characteristic, a selection of metrics from the ISO9126 standard parts 2 and 3 (ISO/IEC 9126-2 and ISO/IEC 9126-3) and from the product requirements was made (e.g. Mean Copies Between Failures). The selection of metrics to be used was mainly based on the fact whether it was possible and easy to measure them, mainly because this was the first experiment with measuring quality characteristics.

Most of the requirements could be used directly from the ISO technical reports and some of them had to be fine-tuned to the project's definitions. An example is the ISO9126 metric Mean Time Between Failures (MTBF). This metric is often used to give an indication of the maturity of the system. For copier/printers the maturity is often indicated by means of the metric Mean Copies Between Failures (MCBF). Therefore a slight change in the metrics could be desirable. Furthermore the product requirements often include quality statements applicable to the product as a whole. As only the controller software was taken into account, these quality aspects needed to be translated for the controller software. E.g. the MCBF is higher for the controller software than for the product, because only failures caused by the controller software should be taken into account. Paper jams are not important when evaluating the maturity of the software.

In total 16 metrics were defined, of which only 1 was defined additional to the internal and external metrics provided in the ISO technical reports. The availability of design documentation was considered to be a good indication of the maintainability. Therefore a metric was added to verify the amount of design documentation that was available versus the amount of

design documentation planned. Examples of the metrics defined can be found in table 2.

Quality characteristic	Sub-characteristic	Metric	Purpose
Functionality	Suitability	*Functional implementation completeness:* Number of missing functions detected during system testing / Number of functions described in requirement specifications.	How many functions have been implemented in relation to the number of functions specified in the requirement specifications?
Reliability	Maturity	*Mean Copies Between Failures:* Total number of copies during system testing / Number of defects, caused by controller software, detected during operation time.	How frequent are the defects of the controller software in operation?
Maintainability	Analysability	*Availability of design documentation:* Available (and approved) design documentation (i.e. SW architecture, top-level design, analysis views, design views and interface specifications) / Identified design documentation.	What's the proportion of design documentation available?

Table 2 Examples of metrics defined

Measurement

It was decided that a hypothetical baseline had to be defined before starting the actual measurement. Our goal was not to measure relative improvement with each test-cycle, but to compare the measured quality against an absolute goal, which was to be reached before the product could be released. Defining a baseline in advance forces people to start discussion when quality targets are not met. If no baseline is defined one is tempted to accept the quality as is, because no reference exists. The product is considered to be 'good enough'.

This baseline was defined by asking a number of experienced people within the project and comparable software projects for an estimate on each metric. These estimates were then averaged. Each estimate was requested to be the 'minimum but sufficient level' for release. E.g., the metric Mean Copies Between Failures was defined to be 100000. This holds that only 1 failure, attributable to the controller software, may occur every 100000 copies.

Decided was to measure all metrics (when applicable) during the system test phase of each incremental development cycle. The result of each development cycle could thus be scored against the defined baseline. In this way, both the improvements per development cycle, as well as the

discrepancy with the 'minimum but sufficient level' could be depicted and reacted upon.

Collection of the metric data during the system test phases implied that some thorough administration was necessary during the test execution. Besides the defects found it was important to administer e.g. the number of copies made, the total down time of the copier/printer and the number of failures successfully restored by the system itself. It was important to clearly instruct the test engineers on what data should be recorded, otherwise important information to compute the metrics could be missing.

After each system test phase the measured values needed to be evaluated to see whether the baseline values defined at the start were still correct. Where necessary the baseline values could be modified, but then at least the discussion took place on the desired quality level. Project management should always have approved changing a baseline.

Real life experience

Until the moment of writing this paper, 3 development cycles have been tested and scored. For one increment, an extra system test cycle was added, leading to a fourth measurement (2 patch). Some results of this increment are presented in the table below.

Quality-characteristic	Sub-characteristic	Metric	Base-line	Value C1	Value C2	Value C2 patch	Value C3
Functionality	Suitability	Functional implementation completeness	0.90	0.91	1.0	1.0	0.82
		Functional implementation correctness	0.80	0.45	0.80	0.84	0.61
Reliability	Maturity	Defect detection	0.75	0.46	0.63	0.63	1.08
		Mean copies between failures	100000	n.a.	93	175	4880
		Test completeness	0.90	0.75	0.78	0.33	0.92
	Recoverability	Mean down time	10 min.	n.a.	5 min.	5 min.	5 min.

Table 3 Measurements results

From the table can be seen that not all metrics were measured for all increments. This is caused by the incremental development methodology used within the project, where it appeared to be impossible to measure e.g. 'Restorability' on release 1, since this functionality was not yet present.

The mean copies between failures is far below the baseline defined. Still this baseline is not adjusted. The low number for MCBF is caused mainly by the fact that the controller software was not yet robust for failures in the

scanner or printer. E.g. a paper jam also resulted in a failure in the controller software. Release 3 was robust for paper jam and one can see that the MCBF increased tremendously.

Interesting to note are the measurements on release 2 patch compared with release 2. The actual use of the software resulting from release 2 was hindered by major instability and performance problems. These problems were resolved, after which measurement 2 patch took place. Although the difference between the metrics of release 2 and release 2 patch is only relatively small (only 9 major defects were solved, approx. 5% of the total amount of defects solved), the users perceived release 2 patch as a much 'better' system. This example shows that the metrics and values indicated in the previous table should be carefully interpreted. When the system only shows a few critical defects in the most used part of the system this system will be of an unacceptable quality level, but the metrics show otherwise. The severity of a defect and its location in the system is not taken into account. So besides the metrics also an evaluation based on common sense has to be made.

For release C3 one can see that the defect detection rate currently is higher than 1, which means that more defects have been found than initially expected. The calculation for the defects expected to be found might be incorrect, which has to be investigated. Literature and metrics of the own organisation were used to derive the number of defects expected to be found.

Conclusion and final remarks

We have now been measuring software quality metrics for less than a year. With more future releases planned (including more extended test periods), our metrics will improve, to the point where we can more clearly use them to set development priorities.

The metrics currently only are used to see what the quality level of the controller software is. The next step will be to submit changes in the development process to improve the quality of the software itself and the process. E.g. if the defect detection rate is lower than expected, as can be seen for releases C1 to C2 patch, but the test completeness is quite high (75% - 78%) it might be necessary to evaluate the effectiveness of the test process. It's of course also possible that the engineering team makes fewer errors as expected.

Furthermore it's still unknown what will happen if the metrics show an insufficient quality level, but the release date has come. How will the organisation react on this and how much importance will it attach to the metrics? This is what still has to be found out.

Up till now the reporting on quality has been received well within the project, both for people involved in the quality model development and those only receiving the test results. The statements on product quality are now based on more than the number of defects and the feeling one has. Besides in the controller software development this method will now also be lifted to project level in order to make statements on the quality level of the copier/printer product as a whole.

As test and quality engineers we're very positive about the ISO9126 approach for defining quality and the questionnaire based method. It gave us a way of defining and reporting product quality in a clear and, quite, unambiguous manner. We've learned a lot and will continue the measurements for the remaining of the project and intend to also use it in future projects.

Authors information

Rob Hendriks
Improve Quality Services B.V.
Waalreseweg 17
5554 HA Valkenswaard
The Netherlands
E-mail: rhe@improveqs.nl

Robert van Vonderen
Océ Technologies B.V.
P.O. Box 101
5900 MA Venlo
The Netherlands
E-mail: lvvo@oce.nl

References

ISO/IEC 9126-1:2000, Information technology – Software product quality – Part 1: Quality model, International Organization for Standardization.

ISO/IEC 9126-2:1999, Information technology – Software product quality – Part 2: External metrics, International Organization for Standardization.

ISO/IEC 9126-3:2000, Information technology – Software product quality – Part 3: Internal metrics, International Organization for Standardization.

Solingen R. van & E. Berghout (1999), The goal/question/metric method, a practical method for quality improvement of software development, McGraw-Hill, UK, ISBN 007-709553-7.

Trienekens J.J.M. and E.P.W.M. van Veenendaal (1997), Software Quality from a business perspective, Kluwer Bedrijfsinformatie, Deventer, The Netherlands, ISBN 90-267-2631-7.

Veenendaal, E.P.W.M. van & J. McMullan (eds.) (1997), Achieving Software Product Quality, UTN Publishers, ‘s-Hertogenbosch, The Netherlands, ISBN 90-72194-52-7.

Appendix A Example product quality characteristics questionnaire

This appendix contains some of the questions of the product quality characteristics questionnaire. For each question the related quality sub-characteristics are indicated.

Process related questions:

What kind of market type is the product oriented towards?
1. Business market
2. Consumer market

(Related to suitability, interoperability, learnability, resource utilisation and time behaviour)

What is the geografic market target?
1. Local
2. Global

(Related to suitability, interoperability, learnability, resource utilisation and time behaviour)

What is the number of products to be sold in a certain market area?
1. 1-1000
2. 1000-10000
3. More than 10000

(Related to suitability and maturity)

User related questions:

What is the average experience of the recognized user groups with regard to the product?
1. More than one year of experience
2. Less than one year of experience
3. No experience

(Related to understandability)

What is the average age of the users?
1. Under 25
2. 25 – 40
3. Older than 40

(Related to learnability and attractiveness)

Software product related questions:

Are there any alternatives to carry on with the activities when the software fails?
1. Yes
2. No

(Related to reliability)

Does the product perform actions without the user intervention?
1. Yes
2. No

(Related to understandability, learnability and operability)

QWE2000 Session 3I

Mr. Massimiliano Spolverini [Italy]
(Etnoteam)

"Measuring And Improving The Quality Of Web Site Applications (3I)"

Key Points

- Web site quality
- Suggested methods
- Techniques to evaluate and improve Web site quality

Presentation Abstract

Up to January 2000 there were approximately 10 million Web sites, 25 million are foreseen by the end of the year 2000 and approximately 100 million for the following year [1]. This explains why competition becomes stronger and stronger in the Web economy era. Visitors can easily switch their attention away from the Web site they are visiting, they feel free to purchase from any supplier no matter how far it can be, because the world is just a mouse click away. VisitorsË choices depend upon features such as the Web site down load time, the ability to easily find the content and services they are looking for, and the trust in the site security and privacy.

The perceived quality of a Web site produces a measurable effect: in the USA in 1999 they estimated losses up to $ 4.4 million due to insufficient Web site quality (Zona Research).

There is a single, simple metric to measure the quality of a Web site. The conversion rate measures the number of visitors who come to a particular Web site within a particular period divided into the number of people who take action on that site, for instance to purchase any of the item on sale [2]. As an average, conversion rates nowadays are in the 3 percent to 5 percent range. For e-commerce Web sites 10 percent conversion rate is considered excellent (i.e. one visitor buys out of 10 who just visit the site).

Merchants will increase their business by increasing the conversion rate. This is why merchants need visitors who are satisfied, remain loyal and therefore create traffic and favourable word-of-mouth.

About the Speaker

Massimiliano Spolverini is Vice President and Chief Consulting Officer of Etnoteam S.p.A., a System Integrator and Consulting firm in the business of the Internet. An

SEI authorized Assessor and ISO Quality Systems evaluator, he is currently involved in the research and application to the Internet technologies of the most accredited frameworks for project management, process and product quality improvement.

ETNOTEAM

THE INTERNET PARTNERS

MEASURING AND IMPROVING THE QUALITY OF WEB SITES

Adriana Bicego, Aldo Maione, Massimiliano Spolverini, Etnoteam S.p.A

Agenda

- Presentation Objectives
- Background
- Toward a definition of Web Quality
- How to build Web site Quality
- Web site assessment and Web Quality

PCTEMP 211-03

PRESENTATION OBJECTIVES

To provide:

- **an overview of background and requirements underlying Web quality**
- **a tentative definition of "Web quality"**
- **an approach to build Web quality**
- **a framework to assess Web quality**

PCTEMP 211-03

BACKGROUND 1

- **A typical Web user clicks away, if he/she cannot boot up a Web page within 8 seconds**
- **At about $362 million per month, perhaps as much as $4.35 billion in e-commerce sales in the U.S. may be lost each year due to unacceptable download speed and resulting user bailout behaviours.**

Zona Research

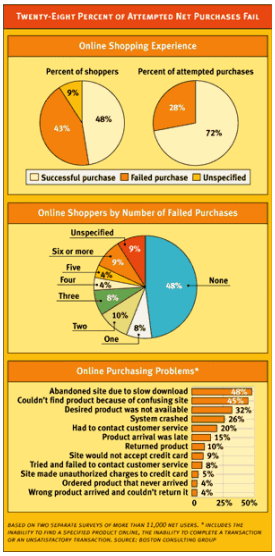
PCTEMP 211-03

BACKGROUND 2

Despite the benefits of shopping online, a staggering 43 percent of US Internet shoppers failed to complete an attempted purchase in the last year:

- **almost half said they abandoned a site due to slow page downloads**
- **another 45 percent ditched Web sites with confusing content or poor navigability**

THE BOSTON CONSULTING GROUP



PCTEMP 211-03

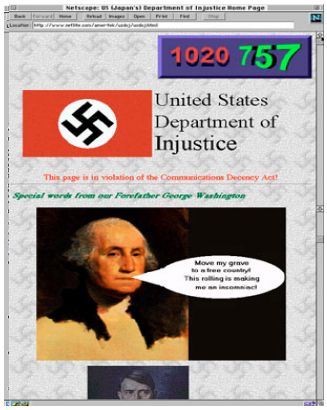
BACKGROUND 3

Trends in Internet security show a remarkable increase of the number of successful attacks reported

US government Web sites were involved in security incidents

- **Department of Justice**
- **NASA**
- **CIA**

Web site security is a key factor to success in the Internet economy



PCTEMP 211-03

BACKGROUND: INTERNET ECONOMY BASIC RULES

- ❑ **Web sites struggle for visitors' time and attention**
- ❑ **The competitors are a click away**
- ❑ **User experience with the site affects buying behaviour**

PCTEMP.211-03

STAKEHOLDERS

Web quality requirements and needs are going to affect more and more decisions and practices of the following players:

- ❑ **Companies (bricks & mortar and Internet pure players)**
- ❑ **Sites users/customers**
- ❑ **Online advertising spenders**
- ❑ **Investors**
- ❑ **Venture capital firms**
- ❑ **Web design agencies**

PCTEMP.211-03

TOWARD A DEFINITION OF WEB QUALITY

Some key questions

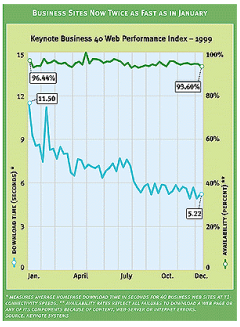
- ✔ *Do site content and functions meet user needs and expectations?*
- ✔ *Can users easily find the site?*
- ✔ *Can users quickly find what they're looking for?*
- ✔ *Does the site communicate effectively?*
- ✔ *Is the site fast and reliable?*
- ✔ *Does the site protect user privacy effectively?*
- ✔ *Does the site assure information security effectively?*

PCTEMP 211-03

IS THE SITE FAST AND RELIABLE ? **1**

In 1999 performance data at 40 major business Web sites showed:

- ❑ an average homepage download time less of 6 seconds
- ❑ an average availability rate of 95%



KEYNOTE
The Internet Performance Authority

➲ Recent researches (*Robert Miller/Jakob Nielsen*) show that 10 seconds is about the limit for keeping the Web user's attention focused on the interaction with the site.

PCTEMP 211-03

IS THE SITE FAST AND RELIABLE ? **2**

Practices which proved to ensure site satisfactory performances (**BEST PRACTICES**)

- To keep the page sizes within 34 KB and to keep objects and multimedia effects to a minimum
- Web pages have to be designed with speed in mind: i.e. to achieve desired response times for various connection speeds:

Modem	34 KB
ISDN	150 KB

- To plan Web infrastructure capacity for future scalability, not just for immediate needs
- ...

PCTEMP 211-03

IS THE SITE SECURE? ❶

A few headlines from CNET news, 25 February 2000:

- *FBI struck by Web attack*
- *Canadian Policy searches ISP for hacker*
- *Hacker discloses new attack software*
- *White House calls for security summit*

...and a few statistics about attacks:

Year	1990	1995	1999	2000
Incidents	252	2412	9859	4226

CERT® Coordination Center

PCTEMP 211-03

IS THE SITE SECURE? ❷

Practices which proved to ensure site security (BEST PRACTICES):

- **APPLICATION SECURITY**
  - Authentication
  - Confidentiality
  - Integrity
  - Non repudiation

Solutions: **digital signature and certificates**

- **NETWORK SECURITY**
  - Access control
  - Traceability

Solutions: **Firewall-based architectures, intrusion detection, penetration test**

PCTEMP.211-03

TOWARD A DEFINITION OF WEB QUALITY

Perceived quality (user experience)

- ☑ Ease of use
- ☑ Navigation
- ☑ Performance
- ☑ Privacy
- ☑ Security

Design and Process Quality

- ☑ State-of-the-art technology solutions
- ☑ Integration with legacy systems
- ☑ Quality of design processes
- ☑ Quality of content management processes
- ☑ Staff and skills used to support the site and content evolution

PCTEMP.211-03

THE USER EXPERIENCE

Web site quality ensures excellent user/customer experience

- ❑ Visitors who get a favorable user experience become loyal users
- ❑ Loyal users generate traffic and favorable word-of-mouth

PCTEMP 211-03

HOW TO ASSESS THE QUALITY OF A WEB SITE?

Senior managers responsible for mission critical Web sites need to assess:

- the **user experience** especially with regard to the site **usability** and **performance** (e.g. response time)
- the compliance with standards, recommendations, laws, rules and codes of practices
- the technologies, processes and skills involved in the Web development and operations
- the policies and solutions implemented to ensure the **site security**

Web quality assessment

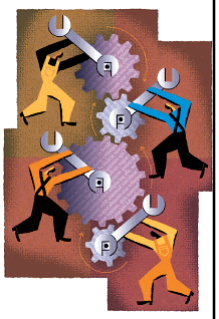
PCTEMP 211-03

WEB QUALITY ASSESSMENT

Includes assessing:

- the SITE ADEQUACY with regard to:
  - **best practices**
  - **standards/rules**
  - **competitors' sites**
  - **best-in-class sites**

BENCHMARKING



- TECHNOLOGIES, PROCESSES and SKILLS used for the **site development, content management and operations**

PCTEMP.211-03

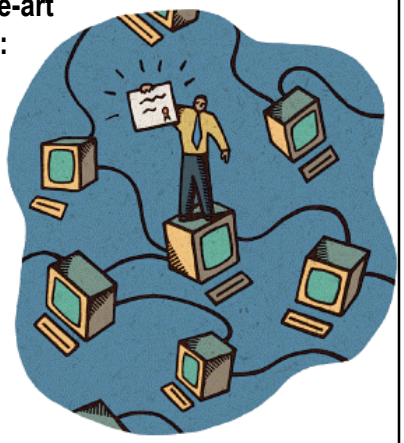
WEB QUALITY ASSESSMENT COMPONENTS

- **USABILITY TESTING**
- **PERFORMANCE Assessment**
- **SECURITY Assessment**
- **TECHNOLOGY Assessment**
- **PROCESS Assessment**

PCTEMP.211-03

USABILITY TESTING

Usability testing, based on state-of-the-art BEST PRACTICES, evaluates the following:



- NAVIGATION
- EXPECTED UTILITY
- CONTENTS
- WORD AND LABEL COMPREHENSION
- COMMUNICATION EFFECTIVENESS
- GRAPHIC ATTRACTIVENESS
- PERCEIVED RESPONSE TIME

PCTEMP-211-03

PERFORMANCE ASSESSMENT

Performance assessment includes:

- actively monitoring the Internet applications during operations to timely detect problems and delayed response time
- evaluating to what extent best practices affecting the site performance are applied

PCTEMP-211-03

SECURITY ASSESSMENT

Security assessment includes:

- **SECURITY AUDIT: complete analyses of the security infrastructure**
- **PENETRATION TEST: attack and intrusion simulation (security scanning)**
- **SECURITY ADVICE: scanning data interpretation to provide detailed advice**

PCTEMP.211-03

21

00 XX

ETNOTEAM
THE INTERNET PARTNERS

AUDITING COMPLIANCE AGAINST STANDARDS

Standards, rules, recommendations and code of practices for e-business sites have been developed:

ISEC eQM2001 rev. 03

GII "Standard for Internet Commerce" ver.1.0

TRUSTe **"**Program principles"

CLICKSURE "Quality Standard for Electronic Commerce" ver 1.4

ALTROCONSUMO "Web Trader Code"

PCTEMP.211-03

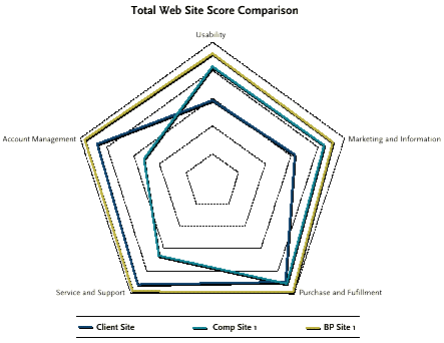
22

00 XX

ETNOTEAM
THE INTERNET PARTNERS

COMPETITIVE BENCHMARKING

The assessment results can be compared to those obtained by the competitors/best-in-class sites



PCTEMP.211-03

TECHNOLOGY ASSESSMENT

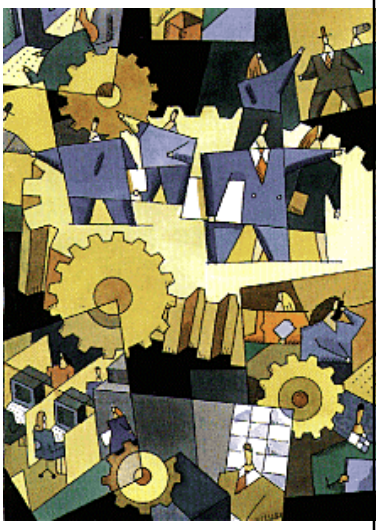
Technology assessment evaluates the site with regard to:

- **HARDWARE/SOFTWARE ARCHITECTURE AND PLATFORMS**
- **INTEGRATION WITH BACK-END/LEGACY SYSTEMS**
- **NETWORK RESOURCES**
- **INTERNET CONNECTIVITY**

PCTEMP.211-03

PROCESS ASSESSMENT

Process assessment evaluates the processes and organisation which support the Web development and operations :



- **CONTENT MANAGEMENT**
- **STAFF E SKILL**
- **WEB DESIGN WORKFLOW**

PCTEMP.211-03

CONCLUSIONS

Web quality is a must to ensure **excellent user experience**

- Visitors who get a favorable user experience become loyal users
- Loyal users generate traffic and favorable word-of-mouth

Drivers which turn visitors into loyal users include:

1. High quality contents, which provide added value to visitors
2. Frequently updated contents
3. Reduced access and download time
3. Trust
5. Ease of use

PCTEMP.211-03

QWE2000 Session 3M

Mr. Kees Hopman [Netherlands]
(IQUIP Informatica BV)

"How to Implement New Technologies? Four Proven Cornerstones for Effective Improvements"

Key Points

- Technology Transfer and Process Improvement
- Customer Focus
- Sandwich Paradigm
- Four Complementing Tracks
- Goal Tracking
- Cornerstone Application Matrix

Presentation Abstract

The presentation deals with the challenges an organisation encounters during a change programme such as the implementation of a new process or new technology. New promising technologies are adopted successfully in one organisation, whilst others still suffer. It's often not the technology that fails but the ability of an organisation to transfer that technology.

Organisations can succeed such implementations by applying four cornerstones for improvements. These cornerstones are:
1. Customer Focus. Focus on your customer: what "improvement" can really help you to make quality products?
2. Sandwich Paradigm. Use both top-down and bottom-up approach: the whole organisation participates.
3. Four Complementary Tracks. Four complementary tracks: blend culture, instrumentation, assurance and fast results.
4. Goal Tracking. Validate your implementation regularly.

The key thing is to balance the four cornerstones in the improvement programme. The Cornerstone Application Matrix (CAM) supports organisations in getting overview and insight to balance their improvement programme.

This presentation describes the characteristics of technology transfer, the four cornerstones and their application. It has been larded and it concludes with work experiences that cover the results of applying the cornerstones.

About the Speaker

Kees Hopman has been active as an IT consultant and improvement consultant since 1987. The last 7 years he has been involved in improvement programmes either as the project leader or as a consultant. The Business Areas have been Banking, Insurance and Telecom.
Kees is an employee of Software Control Quality Care, the unit of IQUIP Informatica B.V. that provides services in the area of quality improvement and quality assurance for processes, projects and products.
In his work Kees has become an advocate of the improvement approach which blends both the instrumental side and the cultural (or human) side of improvement.
Kees is also member of the national competence group "Strategies for Software Process Improvement" of the Dutch Software Process Improvement Network "SPIder" (Stichting SPIder) .

IQUIP

How to Change to New Technologies?

Four Proven Cornerstones for Improvement

Kees Hopman
Change Consultant
IQUIP - The Netherlands

IQUIP
00 nr SC 384.1

IQUIP

Technology Transfer Influences

- Organisational
- Cultural
- Personal
- Communication
- Technology

It is the change ability
that determines dominantly
a successful technology transfer

IQUIP
00 nr SC 384.2

IQUIP

Four Cornerstones for Improvement

- Customer Focus
- Goal Tracking
- Sandwich Paradigm
- Four Complementary Tracks

IQUIP
00 nr SC 384.3

IQUIP

Customer Focus

External view: What service does my customer want?
Internal view: Are we able to provide this service?

IQUIP
00 nr SC 384.4

IQUIP

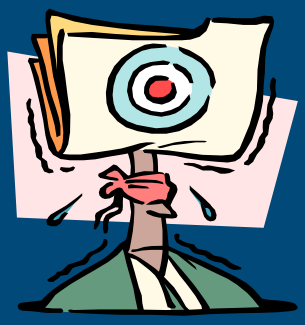
Sandwich Paradigm

Top-Down: vision and structure
Bottom-Up: participation and measures

IQUIP
00 nr SC 384.5

IQUIP

Goal Tracking

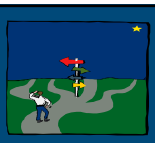


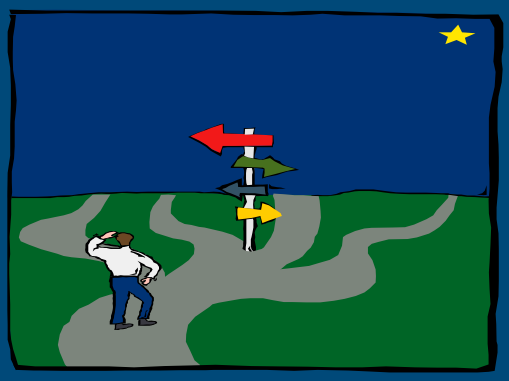
Goal Identification: a goal for everyone
Goal Monitoring and Reporting: insight and show
Goal Evaluation: improve Goal Tracking

IQUIP
00 nr SC 384.6

IQUIP

Four Complementary Tracks



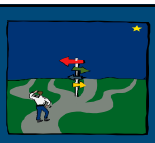


Long and short range measures
Structure and culture measures

IQUIP
00 nr SC 384.7

IQUIP

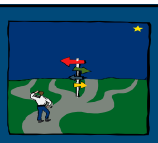
Consolidation Track (1)



- Dimension: Assurance
- Goal:
  - Consolidate present situation
  - Institutionalise new improvements
- Involves:
  - Quality Handbook
  - Audits and Reviews
  - Training and Mentoring
  - Knowledge Management
  - Resource Management

IQUIP
00 nr SC 384.8

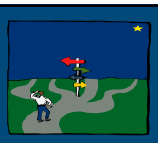
Consolidation Track (2)



- Without consolidation no improvement
- Inflexibility leads also to consolidation

IQUIP
00 nr SC 384.9

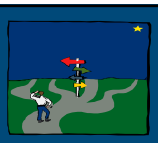
Instrumental Track (1)



- Dimension: Structure improvement
- Goal:
  - Creating instruments to change
- Involves:
  - Identification of new instruments
  - Development of new instruments
  - Pilot and Implementing
  - Training and Mentoring materials
  - Hand over to organisation

IQUIP
00 nr SC 384.10

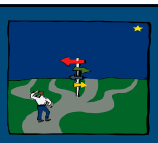
Instrumental Track (2)



- Consolidation Track:
  - Hand over from Instrumental Track to Consolidation Track

IQUIP
00 nr SC 384.11

Culture Track (1)

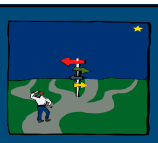


- Dimension: Culture improvement
- Goal:
  - Creating behaviour, attitude and atmosphere for change
- Involves:
  - Identification of current and desired culture
  - Visualise and communicate desired culture
  - Inspiration and motivation
  - Participation model
  - Commitment
  - Enlarge Innovation power

IQUIP
00 nr SC 384.12

IQUIP

Culture Track (2)

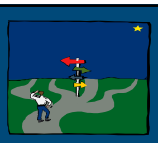


- Consolidation Track:
  - Motivation results in consolidation
  - HR instruments

IQUIP
00 nr SC 384.13

IQUIP

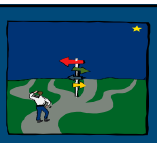
Score Track (1)



- Dimension: Credit
- Goal:
  - Showing intermediate results
  - Convincing people to change
- Involves:
  - Solve acute problems
  - Support vision with evidence
  - Gets improvement team and approach accepted
  - Remove limitations for change

IQUIP
00 nr SC 384.14

Score Track (2)



- Culture Track:
  - Fast and visible results lead to motivation for change

IQUIP
00 nr SC 384.15

Main Pitfalls for Improvement

Perceived pitfalls:

- No focus on customer
- Mobilisation gap
- Too complex solutions
- Culture invisible and intangible
- Flavour of the Day
- Silent success

In terms of Cornerstones:

- Unbalance between Cornerstones

IQUIP
00 nr SC 384.16

Application of Cornerstones

- Use the Cornerstones:
  - Plan Improvements
  - Manage and Control Improvements
  - Provide overview and insight
  - Validate Improvement approach
- Balancing Cornerstones
  - Professional judgement
  - Creating success, avoiding pitfalls
  - Tool: Cornerstone Application Matrix (CAM)
  - Emphasising Cornerstones

IQUIP
00 nr SC 384.17

Case 1: From Scratch

- Improvement goals: Cost reduction and higher productivity
- Solution: Multi-functional test centre
- New technology: TMap®, TPI®, test tools, role model
- Application: CAM scan and monitoring
- Measures:
  - Customer Focus
  - Participation and Motivation
- Result:
  - Test centre established

IQUIP
00 nr SC 384.18

IQUIP

Case 2: Reinforcing the Improvement

- Improvement goals: Higher productivity
- Solution: New technologies
- New technology: DSDM, Powerbuilder/Sybase, CMM
- Application: Reinforcement, CAM scan and monitoring
- Measures:
  - CAM after unsuccessful start reveals unbalance
  - Integrate new technologies
  - Participation and Motivation
- Result:
  - Project on track

IQUIP
00 nr SC 384.19

IQUIP

Summary

- Four proven cornerstones for improvement
  - Customer Focus
  - Sandwich Paradigm
  - Four Complementary Tracks
    - Consolidation
    - Instrumental
    - Culture
    - Score
  - Goal Tracking
- The ability to change determines success

IQUIP
00 nr SC 384.20

"How to change to new technologies?"
Four proven cornerstones for effective improvements

by Kees Hopman

IQUIP Informatica B.V.
Software Control Quality Assurance
P.O. Box 263, 1110 AG Diemen,
The Netherlands

Phone: +31 20 660 66 00
Fax: +31 20 660 66 32

Email: c.r.hopman@iquip.nl

0. Abstract

New promising technologies are adopted successfully in one organisation, whilst others still suffer. It's often not the technology that fails but the ability of an organisation to transfer the technology. Four cornerstones for change are discussed for transferring new technologies or processes. These cornerstones are:

1. Customer Focus
2. Sandwich Paradigm
3. Four Complementary Tracks
4. Goal Tracking

This paper describes these four cornerstones, their dependencies and their application.
It concludes with two work experiences that cover the results of applying the cornerstones.

1. Characteristics of Technology Transfer

> *All models are wrong some are useful*
> *(George Box)*
>
> *All technologies are wrong some are useful*
> *(Kees Hopman)*

Organisations are facing many external and internal influences, under which the introduction of new and promising technologies. It may be technologies like WAP, e-commerce and Internet that support or enable the business processes. Or technologies for software development and software management tooling like code generators and version control systems.
Improvement programmes are initiated to transfer the technologies and to make the promised benefits also profitable for their own organisation. However, various factors influence the likelihood of the technology transfer. Influencing factors such as:

- Organisational factors: the corporate strategy, organisational structure, quality of worklife, rewarding structure, management style;
- Culture factors: encouraging and accepting change, dynamics, change history, the "hidden" organisation;
- Personal factors: professional qualities, faith, perceived necessity, personal win, motivation, trust;
- Communication factors: participation structure, decision making, celebrating successes;
- Technology factors: cohesion and connection with existing technology.

What really determines the success of the technology for an organisation is the ability of the organisation to deal with these influencing factors. Even the 'right' technology for an organisation will not work if the technology transfer isn't successful.

2. Four Cornerstones for Successful Transfer

It is assumed here that a technology transfer takes place during an improvement programme. A lot of improvement approaches have been developed in the past years. Most of these approaches focus on just one or two of the mentioned influencing factors. The reason might be something like “Flavour of the month”, personal preferences, organisational style or perhaps management style.

However, an improvement programme should have a holistic approach. That implies an improvement programme that contains measures for all the influencing factors. To make such improvement programmes an organisation should adapt four cornerstones of change. These four cornerstones are:

1. Customer Focus
2. Sandwich Paradigm
3. Four Complementary Tracks, consisting of:
    A. Consolidation Track
    B. Instrumental Track
    C. Culture Track
    D. Score Track
4. Goal Tracking

Cornerstone 1. Customer Focus

"When customers demand improvement,
organisations will respond."
(Our customer's view)

Customer Focus is about the vision of the organisation. It has an external and an internal view:

- External view: What does my customer want?
  This implies that the organisation must set objectives for serving the customer and must focus on these objectives.
- Internal view: How able is the organisation to provide the service the customer wants?
  This implies that the change capabilities of your organisation should be determined. These change capabilities are necessary to constantly provide the required services.

Customer Focus means that the processes, the technology and the organisation should be focused on the quality of the services and of the products the customer wants, and in terms of the customer. Customer Focus also means that only these processes and technologies are improved and implemented that contribute to the effectiveness and efficiency of your organisation. Accordingly there is a cultural side of Customer Focus: the organisational and personal attitude and willingness to serve the customer.

The main pitfall organisations encounter in the area of Customer Focus is the absence of focus, mainly caused by ‘internal affairs’ or diffuse markets.

Cornerstone 2. Sandwich Paradigm

> *"The mental processes of senior management*
> *have always an advantage on the emotions of their staff.*
> *The pace of improvements puts the adaptability of the staff on test."*
> *(former Vice President)*

Sandwich Paradigm is about the mobilisation of the organisation for changing. It consists of two - complementary - mainstreams: Top-Down and Bottom-Up.

The Top-Down mainstream flows through the organisation from top management to the staff, reflecting the management view. The Top-Down mainstream is the pulling force of an improvement, it sets the direction and preconditions.
Keywords for the Top-Down mainstream are:

- Creation and projection of a Vision;
- Goal setting;
- Creation, communication and implementation of guiding and daily beliefs;
- Determination of limitations to change (budget, capacity, means, organisational structure) and resetting the limitations;
- Commitment to the change and keep the commitment alive and visible;
- Creation of institutionalisation preconditions;
- Walk-the-talk.

The Bottom-Up mainstream flows through the organisation from staff (workers) to top management, reflecting the staff's view. The Bottom-Up mainstream is the pushing force of an improvement, it creates action according the direction and within the limitations.
Keywords for the Bottom-Up mainstream are:

- Participation;
- Define, develop and implement measures;
- Highly motivation;
- Strong institutionalisation;
- Many different views.

Sandwich Paradigm implies that both mainstreams must be present within the organisation. The Top-Down mainstream sets the direction and contour, the Bottom-Up mainstream fills in the contour along the direction.

The main pitfall organisations encounter in the area of Sandwich Paradigm is the *mobilisation gap*. The *mobilisation gap* is the difference in attitude to change of the involved persons. It is caused by absence of awareness, participation, knowledge, ability, communication or motivation.

Cornerstone 3. Four Complementary Tracks

Four Complementary Tracks is about the actual realisation of the change along the four dimensions of change. The four tracks are:
A. Consolidation track
B. Instrumental track
C. Culture track
D. Score track

Track A: Consolidation track

The dimension of the Consolidation Track is assuring the processes, technology and organisation.

The goal of the Consolidation Track is to consolidate the present situation, from a management and control perspective. It is the foundation of each improvement: without consolidation no improvement.

The Consolidation Track involves:
- A storage of
  - Process descriptions,
  - Procedures,
  - Instructions and
  - Templates

  that is
  - Reachable;
  - Traceable;
  - Usable;
  - Maintainable.

  Storage is also known as “Handbook”
- An approach to control and maintain the storage;
- An approach to audit and assess the usage of the processes, procedures and instructions;
- An approach to check the results of the use processes;
- Knowledge Management with respect to
  - Preserving Good Practices;
  - Re-using Good Practices;
  - Information and Training materials.
- Resource Management attributes as:
  - Functional judgements;
  - Role descriptions;
  - Rewarding systems.

The main pitfall organisations encounter on the Consolidation Track is that they can’t keep their “Handbook” simple, lean and mean. Handbooks tend to grow to a Moloch:
- Unreachable: “But it is available on Intranet”
- Untraceable: “Where can I find the process I'm using?”

- Unusable: “We are in step 4.3.0.1.4b, subsection 23”
- Unmaintainable: “I'm sorry, it takes at least 6 months to update your process description.”

Track B: Instrumental track

The dimension of the Instrumental Track is improving from an instrumental viewpoint.

The goal of the Instrumental Track is to reach the improvement objectives by creating new instruments. The Consolidation Track provides the Instrumental Track with input of existing processes, standards, approaches and Good Practices. Reverse, the Instrumentation Track provides the Consolidation Track with new instruments, once they are accepted or implemented.

The Instrumental Track involves:
- Identification of the necessary instruments;
- Definition and development of the instruments;
- Piloting and implementation of the new instruments;
- Providing training and information materials;
- Hand over to the organisation, using the Consolidation Track.

Examples of instruments are process descriptions, procedures, instructions, but also techniques, tooling, training packages, rewarding-mechanisms, career paths.

The main pitfall organisations encounter on the Instrumental Track is to make too complex instruments, too many detailed instructions and too early automation of instruments.

Track C: Culture track

The dimension of the Culture Track is improving from a cultural viewpoint.

The goal of the Culture Track is to reach the improvement objectives by creating an atmosphere for change. The Consolidation and Instrumental Track provide the Culture Track with information and training material and Resource Management attributes. Additional, the Culture Track cares for the right behaviour and attitude to use the new instruments and keep using them.

The Culture Track involves:
- Identification of the current culture;
- Definition of the desired culture;
- Identification of instruments to change the culture (input for the Instrumental Track)
- Identification and planning of the change strategy;
- Define the participation strategy: the roles and responsibilities for the change;
- Creation of a basis for change;
- Creation of momentum for change;
- Inspiration and motivation of people involved.

Typical examples of Culture Track activities are:
- Plan the communication;
- Informing people about the (their) need for change;
- Informing people about the approach;
- Strong participation of people involved in the change;
- Training in process thinking;
- Vision quests;
- Encourage Re-use, or: Discouraging Not-Invented-Here syndrome;
- Teambuilding sessions;
- Informing customers about the new culture;
- Customer participation in definition new culture;
- One-on-one sessions;
- Management coaching;
- Building trust by making and keeping commitments, honesty, learn from mistakes, asking feedback;
- Removing resistance;
- Structurally reward participation in the program and usage of the results of it.

The main pitfall organisations encounter on the Culture Track is the inability of organisations to make the culture tangible and visible. If not tangible, organisations tend to forget the cultural part.
A rather difficult pitfall to overcome is the absence of leaders, either formal managers or natural leaders. Many organisations lack heroes, who will inspire the organisation to change to a new way of living.

Track D: Score track

You must have long range goals
to keep you from being frustrated by short range failures
(Charles C. Noble)

You must have short range successes to build on believe in the long range goals
(Arthur Vermeulen)

The dimension of the Score Track is credit for improvements.

The goal of the Score Track is to reach the improvement objectives by showing frequently intermediate results and celebrating successes. The Score Track is pre-eminently useful to:
- solve a problem immediately;
- solve an acute problem;
- support the management vision with evidence;
- motivate by showing results and successes, both to the improving organisation and to the customer;
- get the improvement team accepted;
- remove limitations for improvements.

It supports the Culture Track by providing evidence to convince non-believers. It provides the Consolidation Track with new instruments, once they are accepted or implemented.

Typical examples of Score Track activities are:
- Remove not used procedures and standards;
- Formalise the frequently used short-cut "procedure";
- Improve what staff think is important to improve.

The main pitfall organisations encounter on the Score Track is Pipedreaming or "Flavour of the Day": leaving the organisation with series of not-ended ad-hoc improvement start-ups.

Four Complementary Tracks concluded

The main pitfall organisations encounter on the Four Complementary Tracks is the unbalance between the tracks. If not balanced well, the change will not occur.
For example: an organisation focuses on the Instrumental and Consolidation Track, neglecting the Culture Track. The result is that the instruments will not be used, although they are technically perfect. The organisation is not able to adopt and use the new technology by heart and mind.

Cornerstone 4. Goal Tracking

Goal Tracking deals with providing quantitative and qualitative insight in the realisation of the improvement objectives. Goal Tracking might be the smallest cornerstone; it certainly is one of the most underestimated and difficult elements of improvement programs.

Goal Tracking is divided into three parts:
- Goal Identification
  Identifying goals and deriving subgoals on each level and for each part of the organisation. For top management, staff, projects, teams, departments.
- Goal Monitoring and Reporting
  Monitoring goals and targets as well as the activities to reach the goals. Reporting results to involved persons and management.
- Goal Evaluation
  Evaluation of the Goal Tracking process to be able to improve this process.

Goal Tracking uses techniques such as the Goal-Question-Metric paradigm or the Quality Function Deployment to identify and monitor the goals and sub-goals.

Goal Tracking has a strong relationship with Customer Focus: to what extent is the organisation supporting the customer? And also with Score Track: Goal Tracking shows intermediate successes!

The main pitfall organisations encounter on Goal Tracking is to track the effort instead of the results.

3. Application of Cornerstones

Many pitfalls for change have been mentioned in this paper. Pitfalls that will be recognised by anyone who is working in the field of change management. Perceived pitfalls are:

- No focus on customer
- Mobilisation gap
- Too complex solutions
- Culture invisible and intangible
- Flavour of the Day
- Silent success

In terms of Cornerstones In terms of Cornerstones these perceived pitfalls could be translated to unbalance between Cornerstones.

It is of great importance to balance and emphasise the cornerstones well and timely before and during the improvement program. Unbalance between cornerstones results in ineffective or unsuccessful programs. Emphasising cornerstones influences the balance and it is the way to manage and control the cornerstones.

Determination of the right balance is one of the most difficult parts. Experience and professional judgement is the answer. One can gain in professional judgement by doing, analysing, learning and relying on feelings.
However, a somewhat simple but effective method to determine the right balance is to over-emphasise a measure, either in practise or in theory. What is the effect of doubling the process audits? What is the effect of more instructions? Or weekly training sessions? Or participation of half the staff? The answers might give you a clue for the right balance.

Organisations use the Cornerstone Application Matrix (CAM) to help them with the application of the cornerstones. The CAM supports them in getting overview and insight.
As an example, the following CAM is filled in for a simplified five-step improvement program that has been reduced to the essence. These five steps are represented by the rows. The columns represent the cornerstones. Each partition lists examples of applicable improvement activities.

Cornerstone Application Matrix	**Customer Focus**	**Sandwich Paradigm**	**Four Complementary Tracks**	**Goal Tracking**
Step1: **Be Aware and** **Be Committed**	• Aware of customer/market • Aware of own performance • Commit to change	• Determine Participation Model • Envision Top-Down • Mobilise Bottom-Up	• Awareness of necessary performance and gap	• Identify Best in Class ranges
Step 2: **Determine Current Status**	• Determine current and future market • Determine own performance on current and future market • Determine Customer Satisfaction	• Determine organisational behaviour • Determine Personal behaviour and Added Value	• Determine ability to consolidate (Consolidation) • Determine present status of instruments (Instrumental) • Determine ability to change (Culture)	• Zero measurement • Identify and assemble historical data

Cornerstone Application Matrix	**Customer Focus**	**Sandwich Paradigm**	**Four Complementary Tracks**	**Goal Tracking**
Step 3: Set Goals	• Set goals related to product, services, markets (external view) • Set goals related to processing and performing services and products (internal view)	• Set Top-Down Targets • Set Individual Targets, derived from or added to the Top-Down Targets	• Determine long range goals • Determine scoring possibilities	• Implement Measurement Program • Make goals quantifiable
Step 4: Improve	• Value operational solutions against customer focus	• Top-Down: Show sponsorship and commitment • Bottom-Up: Participate and Act	• Define operational solutions • Implement • Show Successes • Consolidate	• Track Goals • Feed backwards and corrective action • Feed forward
Step 5: Go to step 2	• Consolidate new customer focus • Validate new Customer Satisfaction	• Consolidate new organisational relations	• Consolidate new processing and technology • Consolidate new culture • Show usage of (improved) instruments	• Validate Results against Goals • Archive Results for future use

In practice, each organisation shall create its own Cornerstone Application Matrix. The advantages are obvious:

- The CAM provides an organisation a tool to balance and emphasise the improvement measures;
- The CAM provides an overview of measures taken for every cornerstone for each improvement step;
- The CAM provides management insight in planning and tracking the application of the four cornerstones.

During the initiation of the improvement programme the CAM is set up. Preferably by the programme manager with top management, middle management and staff. If possible, customers are invited to join. Regularly, f.i. at the end of each step, the same group evaluates and validates the CAM:

- Are the measures taken ready and effective?
- Are the planned measures sufficient?
- Are the planned measures balanced?
- Are there any areas that should be emphasised?

As matter of fact this use of CAM is an example of Goal Tracking.

4. Work Experiences applying the Cornerstones

Two work experiences as an example of organisations that really learned and earned from the improvement cornerstones.

Work Experience 1: The Test Centre

This work experience is an example of where the cornerstones were applied right from the beginning.

Company characteristics: government.
Starting position: several test groups, 4-20 persons, one main application each group, system testing.
Goals: cost reduction, higher productivity, higher effectivity.
Solution: establishment of a multi functional test centre.
New technology: TPI ®[1], TMap®, testtools, role model.

This company wanted to reduce costs and at the same time increase the effectivity and productivity. The solution was to merge several small test groups into a multi functional test centre. This test centre should provide several test services (system testing, system integration testing, acceptance testing, consultancy, test scripts) for several applications on several platforms. In the past a few attempts were made to merge the groups but never succeeded. Often due to lack of top management commitment, diffuse goals and a non-fit in the present culture.
The new responsible top manager decided to launch a new programme: more tests for less cost. A TPI ® assessment revealed strong and weak points of the main test group (20 persons) in the test domain: management, process, tooling and organisation. The TPI assessment report provided the input for the initial programme. A CAM scan validated the initial programme on the change areas. As CAM scan result the initial programme had been completed on several points: multi-level improvement organisation, participation model and on the consolidation, score and culture track.

Other examples of successfully applied improvement areas - related to the Four Cornerstones - were:
Customer Focus:
- Goals and objectives are on demand of the customer;

Sandwich Paradigm:
- Top management commitment and support,
- Bottom Up mobilisation: everyone participated in the working groups to improve the process;

Four Complementary Tracks:
- Consolidation Track: establishment of Quality Handbook, training materials, process- and product audits;

[1] Test Process Improvement (TPI ®) and Test Management approach (Tmap ®) are registered trademarks of IQUIP Informatica B.V.

- Instrumental Track (according to TPI assessment): identification of test types, implementation of defect registration and process, test reference intake;
- Culture Track: adapt and adopt tools of other test groups, coaching, test process mindset, information sessions for (future) customers;
- Score Track: 4 week improvement cycles: every month new small process updates, based on TPI® and TMap®.

Goal Tracking:

- Hour registration and budget control (improved);
- Cost calculation for projects (improved).

During the programme continuous balancing and emphasising resulted in an effective improvement programme.

Work Experience 2: People and Technology but no Process

This work experience is an example where the application of the cornerstones made the difference between success and failure.

Company characteristics: telecommunications.
Starting position: small software development group, +30 persons in two months, large enhancement on one main application.
Goal: higher productivity.
Solution: introduction of new technologies.
New technology: DSDM, PowerBuilder, Sybase, Architecture & Building Blocks, Capability Maturity Model (CMM).

This department wanted to enhance their application. It was not possible - they claim - to make the software in time in the conventional way. New technology would help them. They selected a - for them - new development method (DSDM) and new software development tools (PowerBuilder and Sybase). Their available staff had no experience with both new technology and tools. The department decided to recruit new staff (hired people) and tripled their group in 2 months with 30 more and less experienced people. Training and additional tools had not been provided: the new group should be able to find their way (…).
It is obvious that this group couldn't succeed in this company. The group focussed on the new promising technology but forgot to focus on the customer (they even had problems to identify the customer) and on their approach (about 30!). They made no use of the consolidation track (technology independent tools and standards). Just the right people and the right technology.
After a few months the department started an improvement program - applying the four cornerstones – to make the group and its project successful.

Examples of improvement areas - related to the Four Cornerstones - were:
Customer Focus:

- Identification of the Customer.

Sandwich Paradigm:

- Top Down: management provided resources for improvement and participated in the improvements (project management);
- Bottom Up mobilisation: everyone participated in the working groups to improve the process and to use the technology in the process.

Four Complementary Tracks:

- Consolidation Track: establishment of the new Quality Handbook, derived from the one used for previous releases, process- and product audits;
- Instrumental Track (according to CMM assessment): introduction of software management (CMM), software development method (DSDM), tools (PowerBuilder, Sybase) and organisational standards.
- Culture Track: teambuilding, process thinking and attitude, Customer participation in the improvement program.
- Score Track: 4 week improvement cycles: every month new small process updates.

Goal Tracking:

- Hour registration (improved);
- Product size (FPA);
- Defect registration.

This group managed to improve their process and to adapt the new technology, despite strong (operational) time pressure. Their improvements matched precise with their needs and moreover with the needs of their customer.

Work Experiences Concluded

The organisation in the first work experience had been prevented from another failure. The latter work experience could show us the difference in results between an unbalanced and a balanced improvement program.
Several other successful work experiences can be described here too, discussing the impact of the cornerstones. Also analysing past - and failed - improvement programs reveals major unbalance in applying the cornerstones.
On the other hand, applying the four cornerstones does not guarantee - unfortunately - reaching the improvement goals. But it certainly does prevent organisations from obvious disasters on beforehand (which is sometimes on itself a success). Moreover, the described work experiences give two examples of organisations that really learned and earned from the improvement cornerstones.

5. Advantage and conclusion

Promising new technologies and processes shine seductive and it is difficult for organisations - that have a 'hot pence' burning in their pockets - not to buy these technologies and processes. Unfortunately, after buying an organisation often fails in transferring the technology for its own use. Often, organisations neglect or have been unaware of the fact that technologies are not selling themselves but should be adopted and be adapted. This adopting and adapting process is not a one-dimensional path but deals with various influences and dimensions.

Four cornerstones have been discussed which give organisations an overview in and direction to a technology transfer. These cornerstones also support organisations to manage or control the various influences and dimensions.

Organisations that balance and emphasise the four cornerstones well before and during the technology transfer are able to succeed their improvements. In practice, organisations find it helpful to use the Cornerstone Application Matrix for that purpose.

This paper is concluded with a reminder and small revision of the first quote:

All models are wrong some are useful
(George Box)

All cornerstones are wrong some are useful
(Kees Hopman)

In short: It's not the model, it's the change ability of the organisation that really counts for success.

QWE2000 Vendor Technical Presentation VT3

Mr. David Walker (TBI)

"Effective Requirements Management Using Caliber-RM"

Key Points

- How to improve the requirements management process with an automated, object-oriented approach
- How to facilitate communication and collaboration among the project team
- How to track requirements through the lifecycle for impact analysis

Presentation Abstract

To support an object-oriented approach to requirements management, David Walker will demonstrate Caliber-RM, a collaborative, Web-based requirements management system from TBI that enables organizations to develop higher quality e-business and enterprise applications. By allowing all project stakeholders-including business analysts, product marketing, developers, testers and end users-to collaborate on project requirements, Caliber-RM helps organizations ensure that their applications will meet end-user needs. Caliber-RM facilitates communication among project teams, providing centralized requirement data to distributed team members and allowing documented discussions about requirements and projects. Through lifecycle traceability between requirements and related development and testing tools, project teams are able to understand the impact of potential requirement changes on the project scope, schedule and budget before the changes are accepted. When requirement changes are made, team members are kept up to date with automatic emails notifying responsible individuals of the changes. The demonstration will show how team members can quickly identify potential requirement problems, and manage scope creep through requirement versioning and project baselines. Those in attendance for this session will learn how to better control their requirements definition and management process to increase quality and meet expectations.

About the Speaker

David Walker, Manager, Worldwide Partner Engineering

In his role at Technology Builders, Inc., David Walker is integral in establishing presence and gaining market share for the TBI-Caliber products, with particular

emphasis in European markets.

TBI's first Caliber-RM implementation consultant, David has improved IT initiatives at many Fortune 500 companies through implementation of requirements management methodologies combined with industry-leading automated tool support. He provides pre- and post-sales assistance and mentoring to account managers and customers. As a sales engineer for TBI, David has leveraged his communication and presentation proficiency in applying real world experience to solve prospects' real life challenges. David also is instrumental in ever-developing requirements management course curriculum for TBI's Learning Institute.

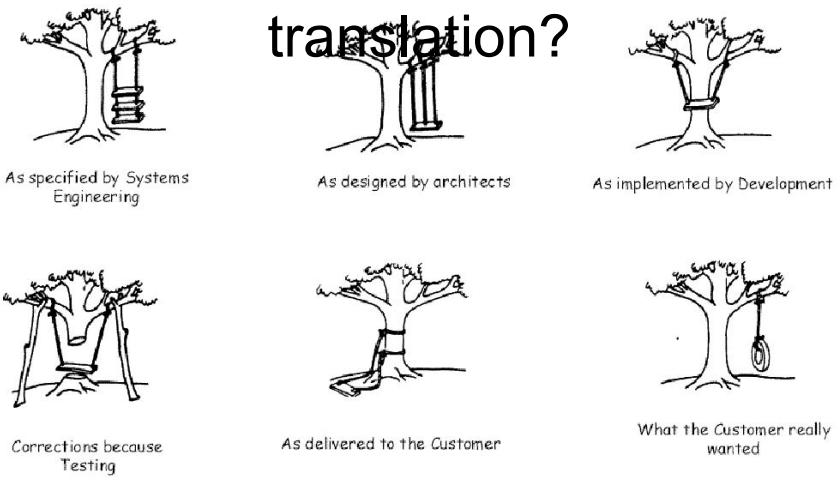
David's extensive background also encompasses project management, process re-engineering, tech courseware development, as well as experience in mortgage banking, investment management, and banking operations. David is a member of SEI and PMI.

Allocating Solutions to Concepts for Successful Product Management

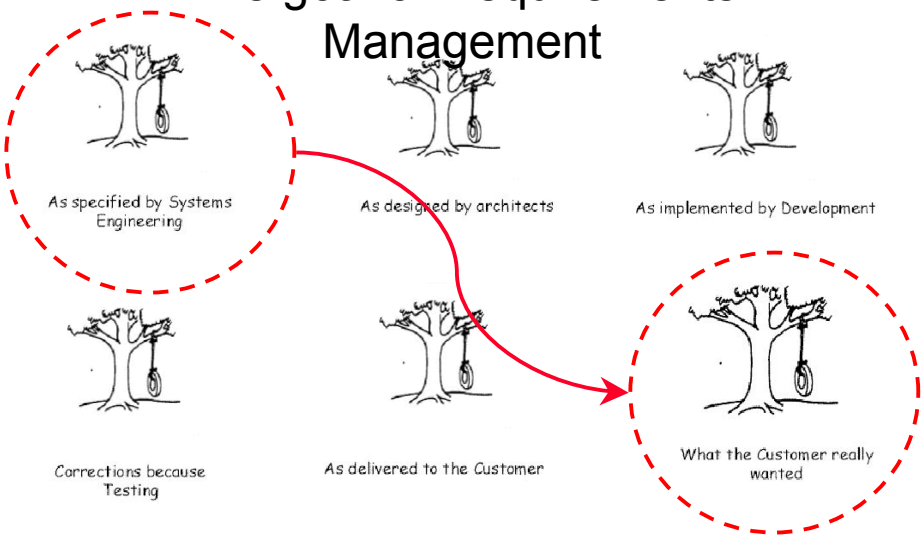
Aligning Technology with Business

CALIBER RM

What gets lost in the translation?



The goal of Requirements Management



What do I need to know?

Why are we doing this task?

What is this component supposed to do?

How will we integrate this?

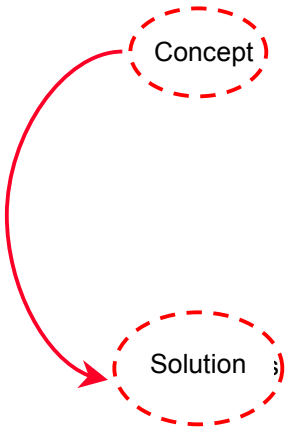
When can I expect this functionality?

Where is this request being fulfilled?

What do I need to know?

Why	are we doing this task?
What	is this component supposed to do?
How	will we integrate this?
When	can I expect this functionality?
Where	is this request being fulfilled?

The whole is greater than the sum of it's parts…

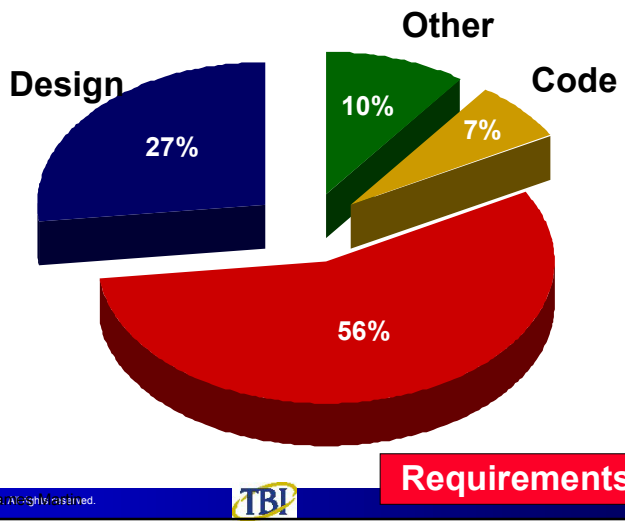


The market is rapidly evolving…

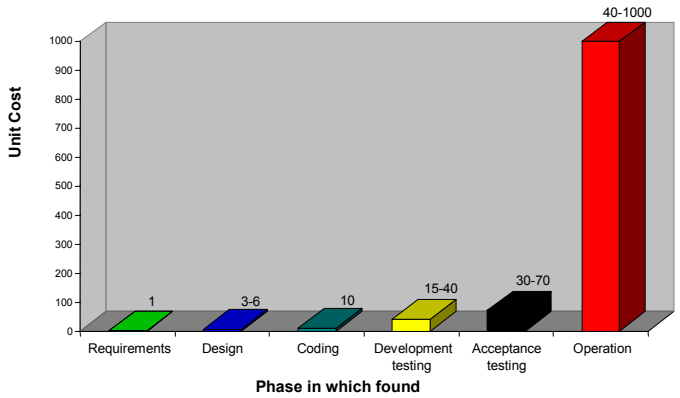
- Software is a relatively immature industry
- The Internet has dramatically changed the marketplace
- The buffer between companies and customers is gone (IE. POS staff)
- Reliable software is a ***must*** to avoid losing customers to the competition
- Reliable software begins with good requirements

RM for Product Management

Distribution of Defects



Relative Cost to Fix an Error*



* A unit cost of one is assigned to the effort required to detect and repair a defect during the requirements phase.
- Gause, Donald and Gerald Weinberg. Exploring Requirements: Quality Before Design.

RM for Product Management

Traditional Requirements Tools

Maturity Level

- Desktop databases
- Word processors
- Spreadsheets
- White boards
- Spiral notebooks
- Post-its
- Napkins

RM for Product Management

CALIBER RM

- True 3-tier client/server/repository
- Internet-based, web-enabled
- Object-centric
- Requirements Management tool
- that supports a collaborative process to fully:
  - define
  - manage and
  - communicate changing requirements

TBI

QWE2000 Session 4T

Lisa Crispin
(iFactor-e)

The Need for Speed: Automating Functional Testing in an eXtreme Programming Environment

Key Points

- Why Testing for XP is Different, challenging conventional wisdom
- How to educate yourself in the eXtreme Programming (XP) methodology
- How to automate functional tests quickly and leverage them to save time

Presentation Abstract

In my two and a half years working in a web environment, where quality and time to market are both essential to success, I've been frustrated by the difficulty in combining these traits within traditional software process. After reading Kent Beck's book, eXtreme Programming Explained, I couldn't wait to try this methodology to enable small teams to deal with short timeframes and changing requirements while still producing high quality software. I recently joined iFactor-e, where we use XP to combine the highest levels of quality and shortest time to market.

Testing in a Web environment can feel like leaping out of a plane. Testing in an XP environment feels like competing in a sky-surfing competition. You have to be better than everyone else, but you don't have much time. You can only hope for a soft landing. While the eXtreme Programming literature (including Ron Jeffries' book, eXtreme Programming Installed), centers around unit and integration testing as part of the XP core process, I felt that functional/acceptance testing from the customer perspective was incompletely defined. The role of the tester in XP is clearly defined - to help the customer choose and write functional tests and to make sure those tests run successfully. The question is, how to do this when the ratio of developers to testers is quite high (8 - 1 is recommended, and we are in a more extreme ratio than that) and the development iterations are so short.

Like an extreme-sports competitor, the XP tester needs courage, speed, stamina and creativity. Working with the developers and with input from an automated test tool vendor, I have developed an approach to designing modularized, self-verifying tests that can be quickly developed and easily maintained. I'll present my basic design and give some examples. I used the test tool WebART, but this methodology should be applicable to any au tomated tool that includes a scripting language.

About the Speaker

I have eighteen years experience in the industry with the last nine in Testing and Quality Assurance. I started out as a programmer and later worked in customer support and QA for large software vendors. In March of 1998, I discovered the world of Web startups, joining TRIP.com as the first test engineer. The challenge of building quality into Web applications while meeting tight development cycles was eye-opening. At TRIP.com, I built a QA department of seven test engineers testing state-of-the-art, first-of-their-kind applications such as flightTracker and intelliTRIP. I felt, however, that we never found a really good process that worked to produce high-quality software in a short amount of time. Missed deadlines were common. Still, we were proud of our accomplishments, as TRIP.com grew to one of the highest-traffic travel Web sites, rated 4.5 out of 5 starts by BizRate and ranked near the top of the Keynote Top 40 websites for performance.

Several developers from TRIP.com left to join a new startup, iFactor-e, devoted to using eXtreme Programming to combine high quality and short time to market to wow the customer. One of these developers loaned me Kent Beck's book. After I read that, I was eager to try XP myself and was fortunate enough to be hired as the first test engineer at iFactor-e in July of 2000. Since then, I have been racing to establish a functional testing methodology that successfully applies the values of XP.

I have given successful presentations at both local and international user and QA conferences to audiences of up to 60 people. I have many years experience training both technical and end users.

The Need for Speed: Acceptance Test Automation in an Extreme Programming Environment

Lisa Crispin, Senior Test Engineer, iFactor-e

Contributors:

Tip House, OCLC Inc.

Carol Wade, TRIP.com

1

What is XP?

XP Values:

- Simplicity
- Communication
- Feedback
- Courage

2

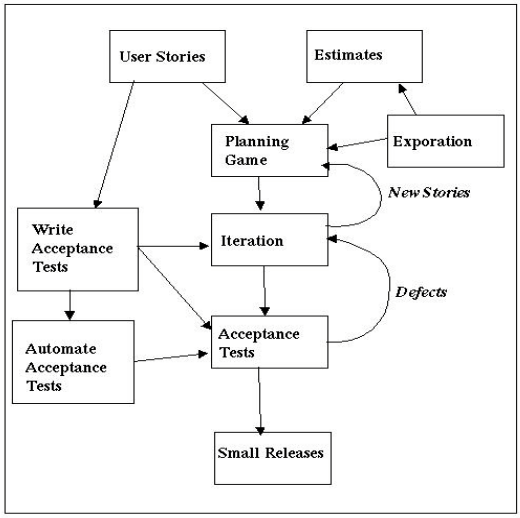
What Makes it Extreme?

Commonsense Practices to Extreme Levels

- If testing is good, everybody tests all the time
- If code reviews are good, pair program
- If design is good, refactor every day
- And so on...

3

iXP Overview



4

Automated XP Testing

This Presentation Will Give Tips On:

- How Testing in XP is Different
- iXP Automated Test Design
- Tools for Writing Acceptance Tests
- Tools for Automating Acceptance Tests
- Tools for Reporting Results

5

XP Practices

At iFactor-e:

- Pair Programming
- Test First, Then Code
- Do the Simplest Thing that Works
- 40-Hour Week

6

XP Practices

At iFactor-e:

- Refactoring
- Coding Standards
- Small Releases
- Incremental planning

7

XP Differences

How is Testing Different with XP?

- Customer can change mind anytime
- Lack of written documents
- Short cycles 1 - 3 weeks
- High developer/tester ratio

XP Differences

How is This OK?

- Unit tests
- Integration tests
- Unit and integration tests must pass 100%
- Customer writes acceptance tests
- Development team assists with test automation

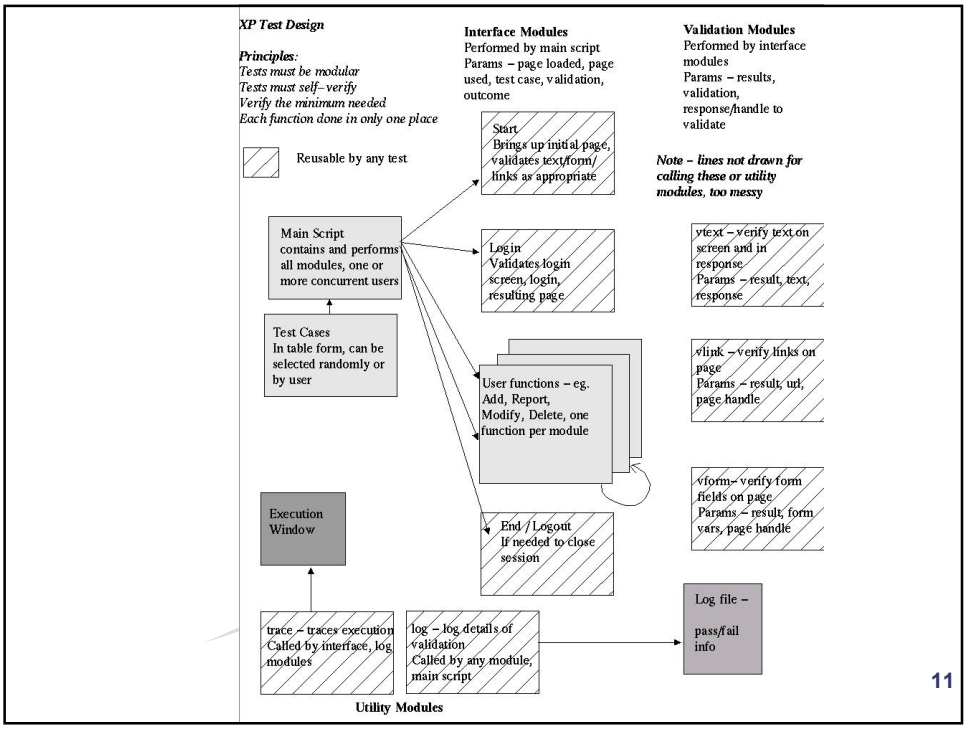
9

iXP Test Automation

Tests must:

- Be modular and self-verifying
- Verify the minimum success criteria
- Contain no duplicate code
- Do the simplest thing that works
- Feature reusable modules

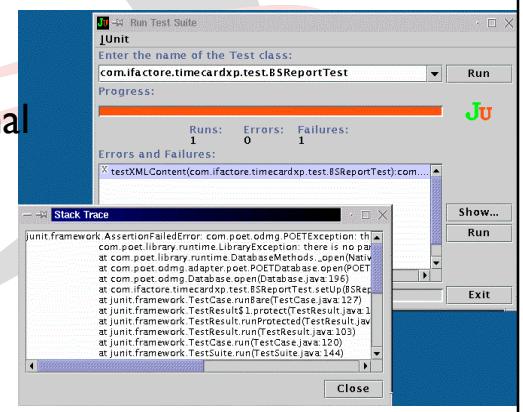
10



iXP Test Tools

Unit Tests

- jUnit test framework
- http://www.junit.org
- Can do some functional
- Run all tests with ant



12

Acceptance Test Automation

XP Principles

- Files full of facts...
- Applied to a system of "fixtures"...
- Encourage change
- Productivity doesn't slow as system grows
- Recorded scripts are murdered by change

13

Acceptance Test Automation

XP Principles

- Customer enters data in spreadsheet, Word table, other format that can be parsed
- Test "fixture" reads data and inputs to application backend
- Record results, write out in html (visual cue such as color)
- Keeps business logic on server side, not in UI

14

Acceptance Test Creation

Spreadsheet Template for Customer Tests

	A	B	C	D	E
1		**Ref #:**	Template for acceptance test: Timecard/QA/AcceptanceTest.sdc		
2		**Iteration:**			
3		**Functionality proven by this test case * is / is not * critical**	**What does this test?** *Example: Tests to make sure that when a time entry record is input, it is saved to the database and the report generated correctly.*		
4		**What to do:**	*Examples given in italics*		
5	**Step**	**Command/URL**	**Action**	**Input Data**	**Expected Output**
6	1	*Access wwwl/timecard/addEntry in browser*	*Select a project, release, iteration, task, and enter duration, date and comments*	*Row 1, columns Project, Release, Iteration, Task, Duration, Date, Comments*	*Selections displaying on screen*
7	2	*Still on wwwl/timecard/addEntry*	*Click the save button*		*Message that data was saved to database*
8	3	*Access wwwl/timecard/generateReports in browser*	*Select a project, start date and end date*	*Row 1, columns Project, Start Date, End Date*	*Report generated – data matches row 1 columns Project, Release, Iteration, Duration, Cost and Total Cost*
9	4	*Repeat steps 1 – 3 with each row in the test case spreadsheet*			

15

iXP Test Tools

Acceptance Test Creation

- Enter test cases directly in XML
- Tool to transfer spreadsheet data to XML

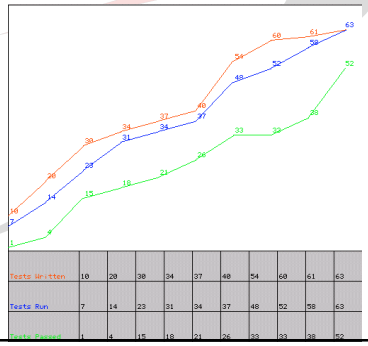
```
<input>
    <loan-amount>1000000000.00</loan-amount>
    <interest-rate>0.5</interest-rate>
    <term-of-loan>1200</term-of-loan>
  </input>
  <output>
    <monthly-payment>A big, fat wad of dough!</monthly-payment>
  </output>
```

16

iXP Test Tools

Automated Testing for Web Applications

- HTTP-based tool - WebART
- Tool inputs test cases to WebART
- Tool to convert result files to charts and graphs



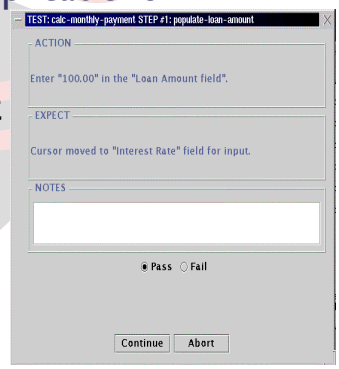
Tests Written	10	20	30	34	37	40	54	60	61	63	
Tests Run	7	14	23	31	34	37	48	52	58	63	
Tests Passed	1	4	15	18	21	26	33	33	38	52	

17

iXP Test Tools

Automated Testing for GUI Applications

- In-house tool "TestFactor-e"
- Prompts user for repeatable test
- Records pass or fail, comments
- Detail and summary reports
- Extending to input automatically



18

XP Acceptance Testing

Acceptance Test Reports

- When run, green bar or background for pass, red for fail - visual feedback
- Allow drill down for detail
- Printed reports posted prominently

19

XP Acceptance Testing

Acceptance Test Reports

- Concrete feedback about current state of system
- Helps team steer the project
- Demonstrates business value of system
- Promotes change

20

The Need for Speed:
Automating Acceptance Testing in an Extreme Programming Environment

Lisa Crispin, Senior Test Engineer, iFactor-e
Contributors: Carol Wade, TRIP.com; Tip House, OCLC.org

"Extreme Programming, or XP, is a lightweight discipline of software development based on principles of simplicity, communication, feedback, and courage. XP is designed for use with small teams who need to develop software quickly in an environment of rapidly changing requirements." Ron Jeffries, http://www.xprogramming.com.

What makes XP Extreme?

As Kent Beck says in Extreme Programming Explained, XP takes commonsense principles and practices to extreme levels. For example: if testing is good, everybody will test all the time (unit testing), even the customers (acceptance testing). Taking anything to extremes can feel scary. While you or I might happily go skiing, we're not likely to ski off the side of a cliff in the manner of Warren Miller. Extreme Programming isn't about taking risks - it's about reducing risks and having fun. It takes courage, but the rewards are immediate.

The XP practices we follow at iFactor-e include:

- pair programming
- test first, then code
- do the simplest thing that works (NOT the *coolest* thing that works!)
- 40-hour week
- refactoring
- coding standards
- small releases
- play the planning game

How is Testing in XP Different?

How does acceptance testing in an XP environment deviate from traditional software testing? First of all, let's look at acceptance testing. XP authors prefer this term to 'functional' testing, as it better reflects how they are written and is more approachable to customers. Acceptance tests prove that the application works as the customer wishes. Acceptance tests give customers, managers and developers confidence that the whole product is progressing in the right direction. Acceptance tests check each increment in the XP cycle to verify that business value is present. Acceptance tests, the responsibility of the tester and the customer, are end-to-end tests from the *customer* perspective, not trying to test every possible path through the code (the unit tests take care of that), but demonstrating the business value of the application.

Should I strap on a helmet and elbow pads?

Testing in an XP environment feels like a run through a half-pipe when you first try it, turning the software development model on its head. The customer is allowed to change her mind anytime. The XP techniques make sure the cost of making changes remain constant throughout the life of project.

Testers may be dismayed at first by the lack of formal written requirements and specifications. To produce small releases very quickly, XP minimizes written documentation. The system is documented through the unit tests, acceptance tests and the code itself. Customers may create mockups of screens and sample reports, but no traditional specifications are written. Design is done primarily with a whiteboard. Collective ownership, promoted by pair programming, reduces the need for written documentation (Which usually is immediately out of date anyway!)

Question: How do you write acceptance test cases without documents?
Answer: You don't. This is the most dramatic way that XP acceptance testing varies from the traditional software development process: In XP, the customer writes the acceptance tests, assisted by the tester.

Other differences between traditional and XP development are more subtle. It's really a matter of degree. XP projects move fast even when compared with the pace at the Web startup where I used to work. It's like running a motocross race when you're accustomed to a street bike. A new iteration of the software, implementing new customer "stories", is released every one to three weeks. The customer must start writing acceptance tests at the beginning of each iteration, as these are the only written "specifications" available. Acceptance tests should run along with unit tests after each integration - which could be several times a day.

From a tester's point of view, the developer to tester ratio in XP looks about as comfortable as street luge. According to Kent Beck, there should be one tester for each eight-developer team. At iFactor-e, the ratio is even higher.

Eeek! Are you SURE protective armor is not required?

Fear not! XP builds in checks and balances that enable a small percentage of test specialists to do an adequate job of controlling quality.

- Becuse the developers write so many unit tests , which they must write before they begin coding - the tester doesn't need to verify every possible path through the code.
- The developers are responsible for integration testing and must run every unit test each time they check in code. Integration problems are manifested before acceptance tests are run.
- The customer is responsible for writing and performing acceptance tests. Naturally, the tester will need to guide the customer in this effort and just as naturally, the customer will soon tire of manual testing and beg for help in the form of test automation.
- The entire development team, not just the tester, is responsible for automating acceptance tests. Developers also help the tester produce reports of test results so that

everyone feels confident about the way the project is progressing.

The roles of the players on an XP team are quite blurred compared with those in a traditional software development process. Thus our iFactor-e XP ("iXP") philosophy is "*specialization is for bugs"*. Here are some of the tasks I perform as a tester:

- Help the customer write stories
- Help break stories into tasks and estimate time needed to complete them
- Help clarify issues for design
- Team with the customer to write acceptance tests
- Pair with the developers to code the application and the test tools
- Pair with the developers to code automated test scripts

Question: Wait a minute. The whole concept of pair programming sounds weird enough. How can a tester pair with a programmer?
Answer: I'm not a Java programmer and our developers don't know the WebART scripting language, but we still pair program. The partner who is not doing the actual typing contributes by thinking strategically, spotting typos and even serving as a sounding board for the coder. This is a fabulous way for developers and testers to understand and work together better. It also gives the tester *much* more insight into the system being coded.

Once you've mustered the courage to jump in to XP, the water's great.

How do I Educate Myself About XP?

Just as you wouldn't attempt to climb Mount Everest without preparing yourself with months of intense training. the XP team needs good training to start off on the right path and stay on it.

Start by reading the XP books. The first book on to be written on XP is Extreme Programming Explained, by Kent Beck. It's a fascinating and quick read. Two new books will be published in the fall of 2000, Extreme Programming Installed, by Ron Jeffries, Ann Anderson, and Chet Hendrickson; and Planning Extreme Programming, by Kent Beck and Martin Fowler.

You can get an overview and extra insight into XP and similar lightweight disciplines from the many XP-related websites, including:
http://www.xprogramming.com
http://www.extremeprogramming.org
http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap
http://www.martinfowler.com

When we at iFactor-e had assembled our first team of eight developers and a tester, we got together and went through Extreme Programming Explained and Extreme Programming Installed as a group, discussing each XP principle, recording our questions (many of them on testing) and deciding how we thought we would implement each

principle. This took several hours but put us all on common ground and made us feel more secure in our understanding of the concepts.

Once your team has read and discussed the XP literature, it's time to get professional training. We hired Bob Martin of ObjectMentor, a consulting company with much XP expertise, for two days of intense training (see www.objectmentor.com for more information). After Bob answered all our questions, we felt much more confident about areas that had previously been difficult for us to understand, such as the planning game, automated unit testing and acceptance testing.

Don't stop there. Talk to XP experts. Look at the Wiki pages and sign up for the egroups. If no XP user group has been formed in your city, start one.

Automating Acceptance Tests

What can you automate?

According to Ron Jeffries, author of XP Installed, successful acceptance tests are customer-owned, comprehensive, repeatable, automatic, timely and produce results that are known to everyone. The "automatic" criterion has given us trouble in some cases, although our goal is to automate whenever it makes sense. Sometimes a mountain bike is the best way up the hill; other times it's easier to get off and walk your bike. For example, we haven't found a cost-effective way to automate Javascript testing. Also we're struggling with how to automate non-Web GUI testing in an acceptable timeframe. Even if we can't automate a test right away, we *can* make it comprehensive, repeatable and timely, and we can publish our results.

Principles of iXP Test Automation

Automated acceptance tests at iFactor-e must meet the following criteria:

- **Modular and self-verifying** to keep up with the pace of development.
- **Verify the minimum criteria for success.** Because the unit tests are comprehensive, we don't need to duplicate it in QA.
- **Perform each function in one and only one place** to minimize maintenance time.
- **Contain modules that can be reused,** even for unrelated projects
- **Do the simplest thing that works. T**his XP value applies as much to testing as to coding.

In addition, the developers try to design the software with testability in mind. This might mean building hooks into the application to help automate acceptance tests. Push as much functionality as possible to the backend, because it is much easier to automate tests against a backend than through a user interface.

iXP Automated Test Design

Appendix A includes a diagram of the design I am using for testing Web applications.

I'm using WebART (see the Tools section below) to create and run the scripts. However, this design should work with any method of automation that permits modularization of scripts. Please see Appendix A for the details of this test design.

Who automates the acceptance tests?

Some sports appear to be individual, when in actuality, they involve a team. Winners of the Tour de France get all the glory, but their victory represents a team effort. Similarly, the XP team may have only one tester, but the entire team contributes to automating acceptance tests. If tools are needed to help with acceptance testing in an XP project, write stories for those tools and include them in the planning game with all the other stories. You'll probably need to budget at least a couple of weeks for creating test tools for a moderately size project.

In the early days of iFactor-e, we initiated a project for the specific purpose of developing automated test tools. This had several advantages, in addition actually producing the tools:

- **Practice with XP** writing stories, playing the planning game, estimating. This gave us confidence in our XP skills that served us future projects.
- **Practice with development technologies.** Developers could experiment with different approaches and get experience with new tools. For example, the developers investigated in advance the advantages of using a dom versus a sax parser on the XML files containing customer test data. Doing this in advance gave us more time to experiment and research technologies than we might have had later with a client project.
- **Mutual understanding.** The team tasked with producing an acceptance test driver consisted of only four members and me, so I was called on to pair program. This exercise gave me insight into how tough it is to write unit tests, write code and refactor the code. The developers gave a lot of thought to acceptance testing and we had long discussions about what the best practices would be. This is a great foundation for any XP team.

Tools

Sky surfers don't leap out of the plane wearing any old parachutes purchased from a discount store. They look for state-of-the-art harness and container systems, main and reserve canopies, helmets and goggles, even altimeters, all designed with their particular needs in mind. XP testers need a good toolbox too, one containing tools designed specifically for speed, flexibility and low overhead.

I've asked several XP gurus, including Kent Beck, Ward Cunningham and Bob Martin, the following question: "What commercial tools do you use to automate acceptance testing?" Their answers were uniform: "Grow your own". Our team extensively researched this area. Our experience has been that we are able to use a third-party tool for Web application test automation, but we need homegrown tools for other purposes.

For **unit testing**, we use a framework called jUnit, which is available free from http://www.junit.org. It does an outstanding job with unit tests. Even though I am not a Java programmer, I can run the tests with jUnit's TestRunner and can even understand the test code well enough to add tests of my own. It's possible to do some functional tests with jUnit. Some XP teams use this tool for automating acceptance tests, but it cannot test the user interface. We didn't find it to be a good choice for end-to-end acceptance testing.

Tools for Creating Acceptance Tests

Some XP pros such as Ward Cunningham advocate the use of spreadsheets for driving acceptance tests. This isn't a new idea. We want to make it easy for the customer to write the tests, and most are comfortable with entering data in a spreadsheet. Spreadsheets can be exported to text format, so that you and/or your development team can write scripts or programs to read the spreadsheet data and feed it into the objects in the application. In the case of financial applications, the calculations and formulas your customer puts into the spreadsheet communicate to the developers how the code they produce should work.

At iFactor-e, we provide a couple of ways for the customer to enter acceptance test cases. Sometimes they work with me to enter test data and test case actions directly into an XML format that is used by our acceptance test driver. If they prefer to use a spreadsheet, we simply convert the spreadsheet into XML format later. We're currently working to make these methods more user-friendly. See Appendix B for a sample acceptance test spreadsheet template.

Appendix C shows a ***partial*** excerpt of a sample XML file used for acceptance test cases. The customer enters a description of the test, data and expected output, steps with actions to be performed and expected results.

Automated Testing for Web Applications

Test automation is relatively straightforward for Web applications. The challenge is creating the automated scripts quickly enough to keep pace with the rapid iterations in an XP project. Like a motocross racer, I'm zipping down hills and slogging through mud, trying to keep up with the pack of developers. For that extra burst of speed, I use WebART (www.oclc.org/webart), an inexpensive HTTP-based tool with a powerful scripting language. WebART enables me to create modularized test scripts, creating many reusable parts in a short enough timeframe to keep up with the pace of development. Javascript testing presents a bigger obstacle. We test it manually and carefully control our Javascript libraries to minimize changes and thus the required retesting. Meanwhile, we continue to research ways of automating Javascript testing.

Our developers wrote a tool to convert test data provided by the customers in spreadsheet or XML format into a format that can be read by WebART test scripts so that we can

automate Web application testing. Even small efforts like this can help you gain that competitive edge in the speedy XP environment.

Automated Testing for GUI Applications

Test automation for non-HTTP GUI applications has been more of an uphill climb. You can travel faster in a helicopter than a mountain bike, but it takes a long time to learn to fly a helicopter; they cost a lot more than a bicycle and ou may not find a place to land. Similarly, the commercial GUI automated test tools we've seen require a lot of resources to learn and implement. They're budget breakers for a small shop such as ours. We searched far and wide but could not come up with a WebART equivalent in the GUI test world. JDK 1.3 comes with a robot that lets you automate testing of GUI events with Java, but it's based on the actual position of components on the screen. Scripts based on screen content and location are inflexible and expensive to maintain. We need tests that give the developers confidence to change the application, knowing that the tests will find any problems they introduce. Tests that need updating after each application change could cause us to lose the race.

We felt that the most important criteria for acceptance tests is that they be repeatable, because they have to be run for each integration. We decided to start by developing our own tool, "TestFactor-e", that will help customers and testers run manual tests consistently. It will also record the results. We're now enhancing this tool to feed the test data and actions directly into application backends in order to automate the tests.

Reports

Getting feedback is one of the four XP values. Beck says that concrete feedback about the current state of the system is priceless. An extreme skier constantly monitors snow conditions, the course, his speed, the state of his equipment, all while keeping an ear out for the avalanche that may be coming along behind him. He accommodates these factors with changes in speed, trajectory and position. The XP team needs a constant flow of information to steer the project, making corrections in mid-course just as the skier would. The team's continual small adjustments keep the project on course, on time and on budget. Unit tests give programmers minute-by-minute feedback. Acceptance test results provide feedback about the "Big Picture" for the customer and the development team.

Reports don't need to be fancy, just easy to read at a glance. A graph showing the number of acceptance tests written, the number currently running and the number currently succeeding should be prominently posted on the wall. You can find examples of these in the XP books. Our development team wrote tools to read result logs from both automated tests and manual tests run with "TestFactor-e". These tools produce easy-to-read detail and summary reports in HTML and chart format.

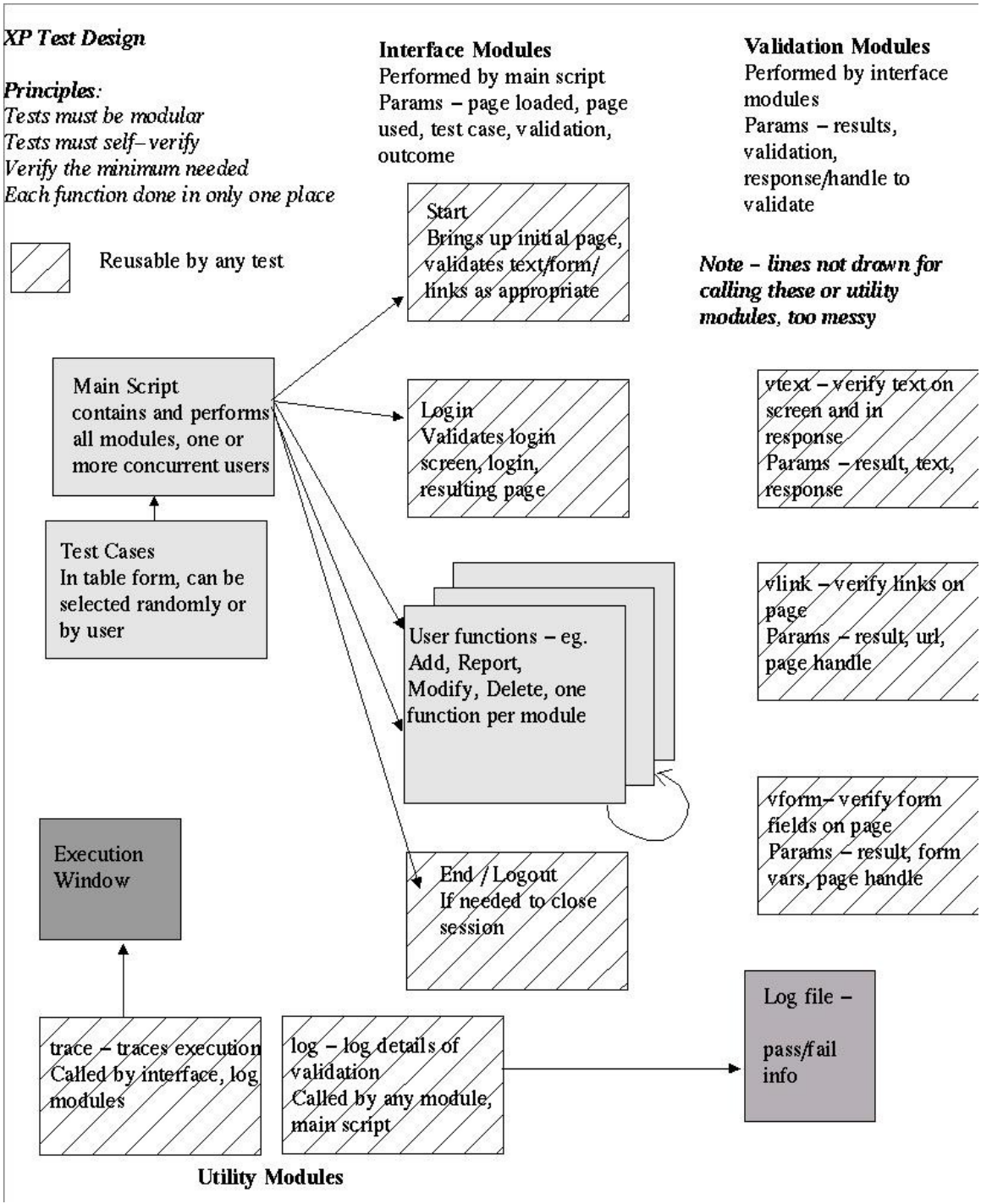
With all this feedback, you'll confidently deliver high-quality software in time to beat your competition. You'll meet the challenges of 21st century software development!

Appendix A: Test Design Description

Because this paper is so long I am not going to include code examples here. Contact me at lisa.crispin@ifactor-e.com if you would like some sample WebART scripts.

Main Script

The **main module** calls the supporting module to perform a typical user scenario and to validate the system responses at strategic points along the way. A basic user scenario is created for each customer story. The **supporting modules** are divided into several groups based on their function. These modules are described below.



The main module passes to the supporting modules test cases from tables of test data along with other parameters.

Sample basic user scenario:

Action	***Minimum Passing Criteria***
1. Go to Add Entry page.	Browser challenges for authentication.
2. Login.	Valid userid/password does not get error.
3. Add a time entry using selected test case (may be done with multiple users).	Form fields are on page at start. After submit, still on Add screen (checks by text and forms.
4. Go to Generate Reports page.	Browser challenges for authentication (Currently, because these are two separate scripts, could be combined).
5. Login .	Valid userID/password does not get error.
6. Generate a report for the date range specified for the test case added in Action 3.	Form fields are on generate reports page After submitting, report contains correct text, correct links back to generate page, and contains the userID, release and iteration from the test case added in Action 3.
7. Log out.	Log out successful message displayed.

Interface Modules

Called by the main script module (and possibly other interface modules) to perform user functions and to validate the correct system response. Some of these modules such as start, login and exit can be used by multiple tests for the same application. The main module passes **parameters** to the interface modules as follows:

page loaded - An *output* parameter; it receives the value of the page loaded by the module. For example, the start module loads the addEntry.jsp page.

page used - An *input* parameter; it is the handle of the page used by the interface module to know what page to load.

test case data - An *input* parameter; it is data to be parsed by the interface module and used for input or validation. For example, for a login module, the test case consists of an *userid* and *password* .

validation clause - Tells the interface module how much validation to perform. A value of STD indicates to do only the minimum validation; extended validation options may be specified and added to STD.

outcome - An *output* parameter that receives a value of **PASS** or **FAIL** indicating the overall outcome of the call.

Additional parameters may be used if needed (eg., more than one page needs to be loaded).

Validation Modules

Called by interface modules to check for specific conditions in a system response and return a pass or fail condition. The validation modules in turn call **utility** modules to record the results. Parameters are:
results - An *output* parameter which returns **PASS** or **FAIL** to the calling module

controls - An *input* containing values that control how the validation is done, specific to each validation module.

response or handle - An *input* parameter containing either the system response to be validated or the handle of the page into which the response was loaded.

Currently, there are three validation modules that can be used by any test:
vtext validates that a response contains specified text. Parameters are:

result - PASS or FAIL

text - the text that must be present for the validation to pass

response - the system response to check for the text string.

vlink validates that a page contains a specific link. Parameters are:

result , *urlMatch* - the value which must exist in a link in the page whose handle is in pPage

pPage - the **page handle** of the page being validated.

vform validates that a page contains a specified form. Parameters are:

result , *formvars* - one or more required variables for the form - if any are missing, the validation fails.

pPage - the **page handle** of the page being validated.

Utility Modules

Currently there are two utility modules which can be used by any test:
trace - Displays execution tracing information in the WebART execution window. Called by interface and log modules. Without going into detail of the many parameters, they reveal who called it and what happened.
log - Records validation outcomes in a log file. Parameters are:

type - detail or summary, *outcome* - PASS or FAIL

validation - describes the validation performed.

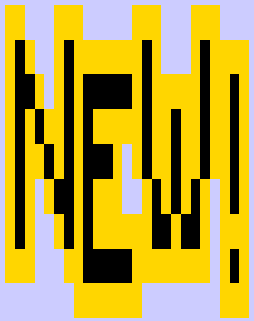
Appendix B: Sample Acceptance Test Spreadsheet Template

	A	B	C	D	E
1		**Ref #:**	Template for acceptance test: Timecard/QA/AcceptanceTest.sdc		
2		**Iteration:**			
3		**Functionality proven by this test case * is / is not * critical**	**What does this test?** *Example: Tests to make sure that when a time entry record is input, it is saved to the database and the report generated correctly.*		
4		**What to do:**	*Examples given in italics*		
5	**Step**	**Command/URL**	**Action**	**Input Data**	**Expected Output**
6	1	*Access wwwltimecardladdEntry in browser*	*Select a project, release, iteration, task, and enter duration, date and comments*	*Row 1, columns Project, Release, Iteration, Task, Duration, Date, Comments*	*Selections displaying on screen*
7	2	*Still on wwwltimecardladdEntry*	*Click the save button*		*Message that data was saved to database*
8	3	*Access wwwltimecardlgenerateReports in browser*	*Select a project, start date and end date*	*Row 1, columns Project, Start Date, End Date*	*Report generated – data matches row 1 columns Project, Release, Iteration, Duration, Cost and Total Cost*
9	4	*Repeat steps 1 – 3 with each row in the test case spreadsheet*			

Appendix C: Partial Excerpt of XML Template for Acceptance Test Cases

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

```
<!DOCTYPE at-test SYSTEM "at-test.dtd" [
  <!ELEMENT input ANY >
  <!ELEMENT loan-amount ANY >
  <!ELEMENT interest-rate ANY >
  <!ELEMENT term-of-loan ANY >
  <!ELEMENT output ANY >
  <!ELEMENT monthly-payment ANY >
]>

<at-test name="calc-monthly-payment" version="1.0" severity="CRITICAL">

  <at-project>mortgage-calc</at-project>

  <at-description>
    Enter loan amount, interest rate, term of loan (in months)
    to calculate monthly payment.
  </at-description>

  <at-data-sets>
    <at-struct id="values">
        <input>
        <loan-amount>1000000000.00</loan-amount>
        <interest-rate>0.5</interest-rate>
        <term-of-loan>1200</term-of-loan>
      </input>
      <output>
        <monthly-payment>A big, fat wad of dough!</monthly-payment>
      </output>
        </at-struct>
  </at-data-sets>

  <at-plan>

    <at-step name="populate-loan-amount">
      <at-action>
        <at-text>Enter "{0}" in the "Loan Amount field".</at-text>
        <at-value dset="values" select="/input[2]/loan-amount"/>
      </at-action>
      <at-expect>
        <at-text>Cursor moved to "Interest Rate" field for input.</at-text>
      </at-expect>
    </at-step>

   </at-plan>

</at-test>
```



QWE2000 Session 4A

Mr. Steve Littlejohn [UK]
(SIM Group Limited.)

"Test Environment Management -- A Forgotten Basic"

Key Points

- Test Environments are a fundamental requirement for good testing. Their role in testing methods and practices is misunderstood.
- The move or integration of Legacy systems to other platforms has complicated rather than improved the use of test environments - E-commerce has added to this complexity.
- Data requirements fall in the gap between configuration management and test management. Test environments need control, management principles, disciplines from cradle to grave.

Presentation Abstract

This presentation is drawn from an extensive knowledge and background in test environment use and requirements. All to often these valuable test assets are ignored, abused or blamed for the poor quality of testing undertaken. This is not limited to a particular platform or type of system, it is a problem over all testing from Legacy to E-commerce. The presentation is based upon customer experiences and uses detailed examples to show how the test environments directly affect the effectiveness of testing and testing practices. It also explains how this could have been prevented and how to implement test environment management.

Beginning with actual examples of the issues with test environments and their affect on testing, the case for test environment management will be built. Otherwise good solid testing practices have failed against the need for a test environment to test in. Plans, scripts and data all tend to be managed and knonwn, test environments are seen as the painful part of testing.

Legacy systems were problematic in terms of test environments and downsizing became fashionable for many reasons. However, in the next examples this is shown not to have improved the test environemnt situation but added extra complexity to the environment requirements. The recent rise in E-commerce has added to the burden of test environments.

Version Control is placed over times in environments, but not necessarily on the test environment objects. Configuration management requirements are highlighted by detailed examples explaining why this is a necessity for test environments.

Data requirements are also largely ignored until the test cases demand that data is made available. A search for data ensues or a creation process takes place - all of which takes it's toll on the test environment. The data issues are explored with detailed examples to illustrate how this vital component is underused.

To implement or renew test environmen management a course of action needs to be in place. This is a suggested solution in order to achieve managed, controllable test environments.

About the Speaker

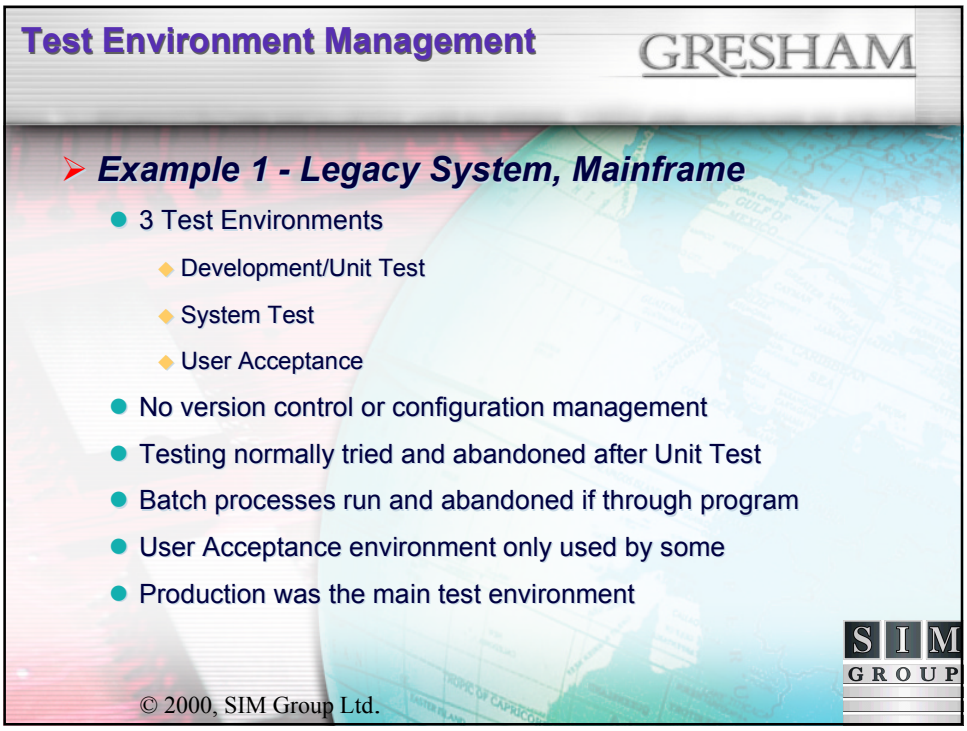
Steve Littlejohn has been involved in the IT Industry for about 20 years. He has worked in most IT roles from operations through to project management via programming, analysis and design before specialising in testing for the past eight years. With SIM Group he is a Senior Consultant and has worked on testing projects for a variety of clients within many industry sectors. He has a special interest in the techniques of test environment management and automation and their use with automated testing.

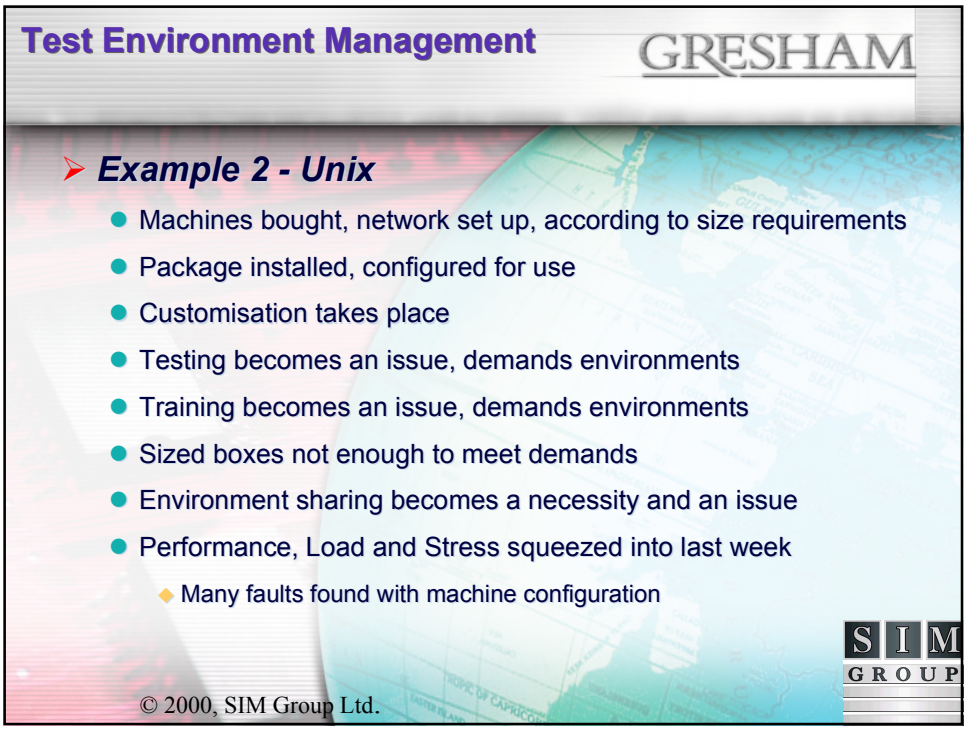

GRESHAM
Test Environment Management
A Forgotten Basic
Steve Littlejohn
steve@simgroup.co.uk


Test Environment Management
GRESHAM
Agenda
Test Environment Management Disasters
Test Environments and the brave new World
Configuration Management - needs and wants
Test Environments vs Data Requirements
Test Environment Management - gaining control
Implementing Test Environment Management
Summary
SIM GROUP
© 2000, SIM Group Ltd.


GRESHAM
Test Environment Management - Disasters, an illustration of what can happen….
S I M GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Why Disaster?
A good environment is a fundamental requirement
For most any old environment will do
Developers do care but put up with what is available
Legacy systems on Mainframe always difficult
Tend to have been grown organically with applications development
Application integration and development “too involved and slow”
Environment maintenance ignored except by necessity
Unix, eCommerce - requirements “bolted on”
Environments available until live then disappear
Test Environments generally do not match target
S I M GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Example 1 - Legacy System, Mainframe
3 Test Environments
Development/Unit Test
System Test
User Acceptance
No version control or configuration management
Testing normally tried and abandoned after Unit Test
Batch processes run and abandoned if through program
User Acceptance environment only used by some
Production was the main test environment
S I M GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Example 2 - Unix
Machines bought, network set up, according to size requirements
Package installed, configured for use
Customisation takes place
Testing becomes an issue, demands environments
Training becomes an issue, demands environments
Sized boxes not enough to meet demands
Environment sharing becomes a necessity and an issue
Performance, Load and Stress squeezed into last week
Many faults found with machine configuration
S I M GROUP
© 2000, SIM Group Ltd.

Test Environment Management

GRESHAM

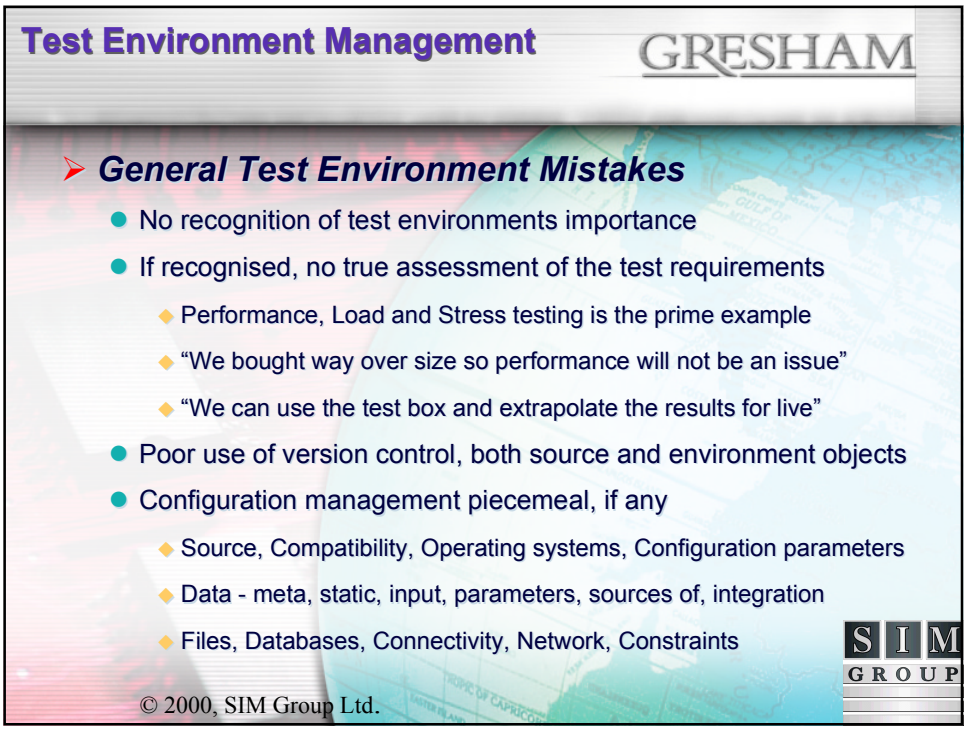
- ***Example 3 - eCommerce***
  - Requirement is to integrate to current platforms and data
  - Unix box sized and purchased
  - Application development partly outsourced
  - Security becomes an issue, cannot be tested due to environment
  - Performance, Load and Stress not seen as a requirement
  - Testing becomes nightmare
    - Integration to shared test environments reduces time available
    - Data available not good enough for application
  - Final realisation that different box required for live environment

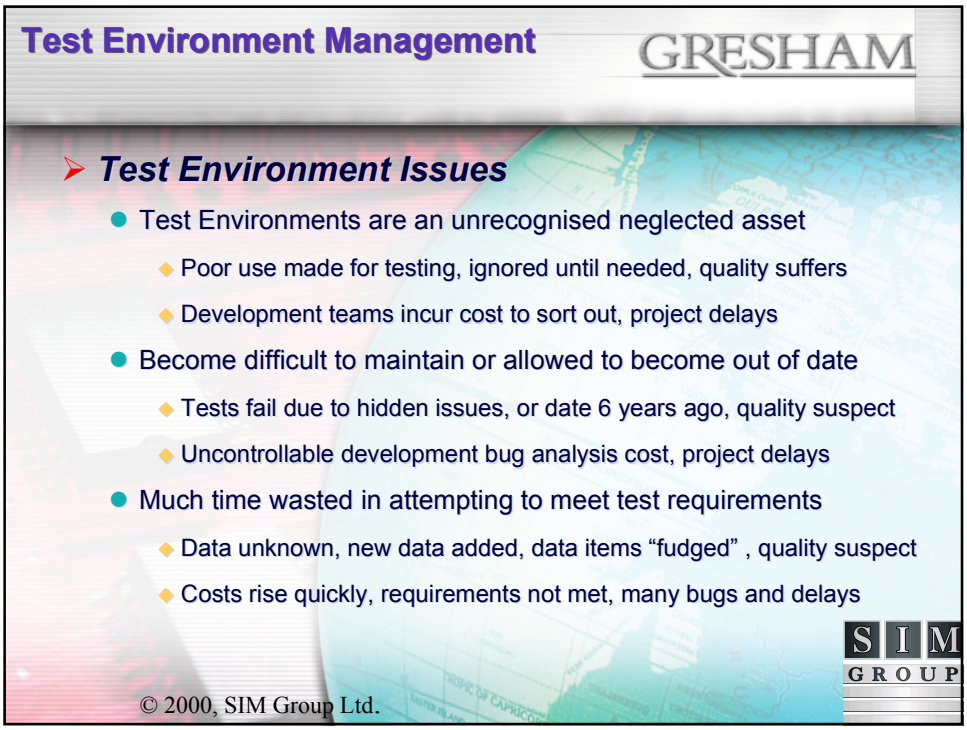
SIM GROUP

GRESHAM

People tend to overly concentrate on hardware and infrastructure as opposed to configuration management and data.

SIM GROUP

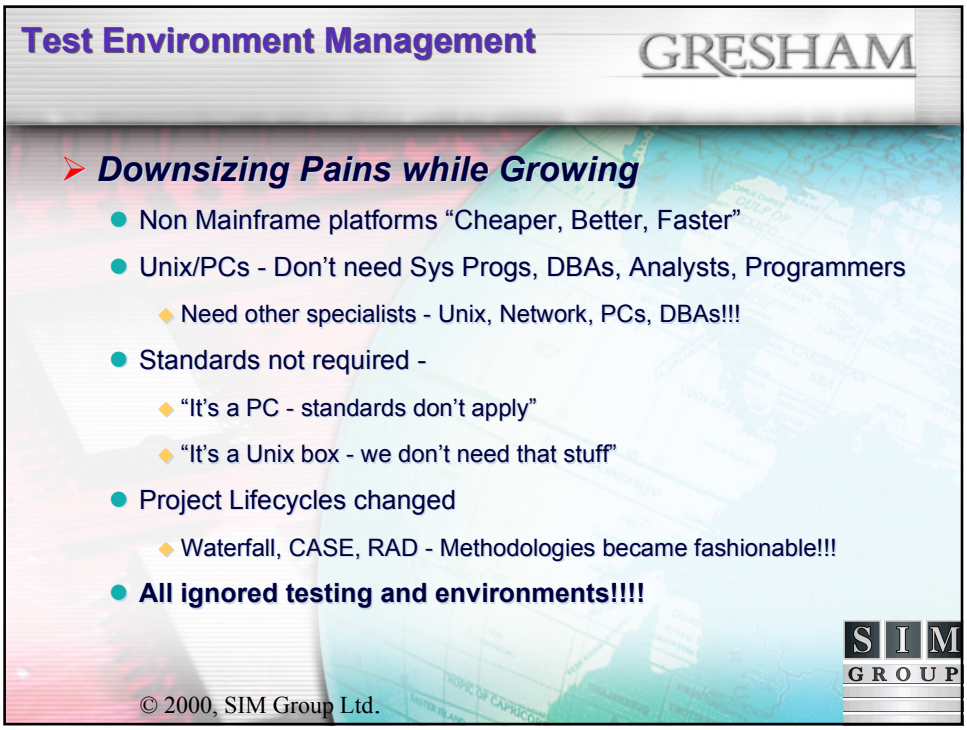

Test Environment Management
GRESHAM
General Test Environment Mistakes
No recognition of test environments importance
If recognised, no true assessment of the test requirements
Performance, Load and Stress testing is the prime example
"We bought way over size so performance will not be an issue"
"We can use the test box and extrapolate the results for live"
Poor use of version control, both source and environment objects
Configuration management piecemeal, if any
Source, Compatibility, Operating systems, Configuration parameters
Data - meta, static, input, parameters, sources of, integration
Files, Databases, Connectivity, Network, Constraints
S I M
GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Test Environment Issues
Test Environments are an unrecognised neglected asset
Poor use made for testing, ignored until needed, quality suffers
Development teams incur cost to sort out, project delays
Become difficult to maintain or allowed to become out of date
Tests fail due to hidden issues, or date 6 years ago, quality suspect
Uncontrollable development bug analysis cost, project delays
Much time wasted in attempting to meet test requirements
Data unknown, new data added, data items "fudged" , quality suspect
Costs rise quickly, requirements not met, many bugs and delays
S I M
GROUP
© 2000, SIM Group Ltd.

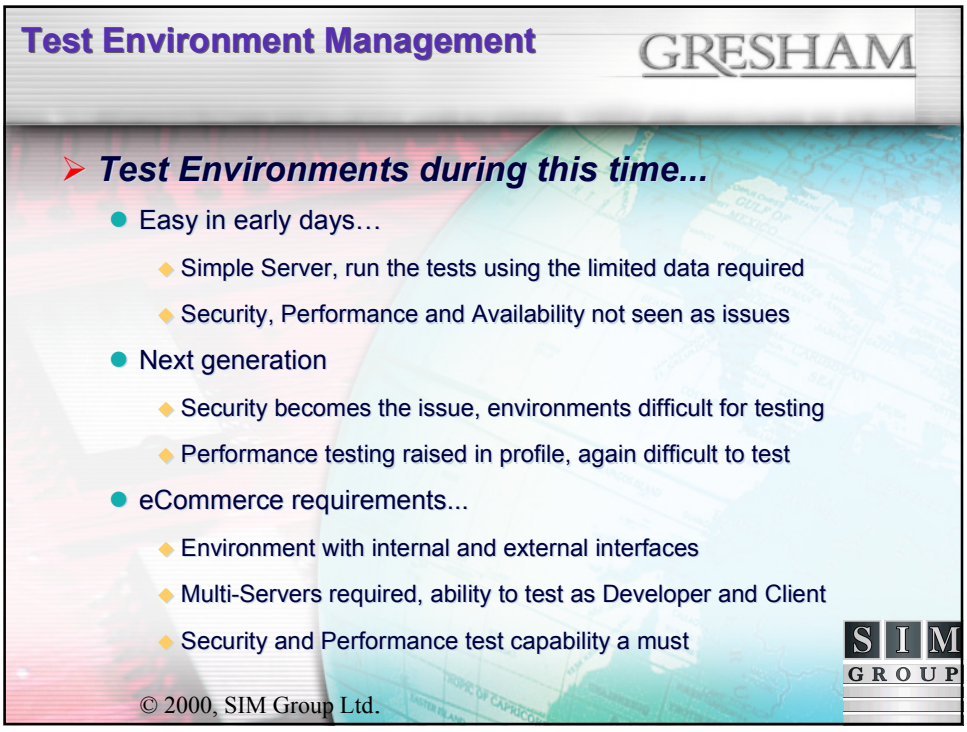

Test Environment Management
GRESHAM
Test Environment Issues
Package data integration not always known
Lack of knowledge leads to poor test requirements, quality poor
Package customisation error rise, increased development cost
Sizing only recognises production needs, testing an afterthought
Test environments shared, tests and time impacted, quality suffers
Testing "squeezed" to meet deadlines, heavy cost of Live bugs
Performance, Load and Stress not considered important
Hardware test requirements ignored as well Software, quality ????
If all configured correctly….timebomb ticking…..Cost of Failure???
S I M GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Test Environment Issues
Small sized environments restrict tests possible
Not always true, normally filled with previous data, quality suspect
Best use not made, much data development at extra cost, delays
Internal or external system connectivity not available
Tests "checked" by file formats or similar, quality suspect
Problems arise, extra development costs occur, many Live bugs
Automation fails most often due to the test environment
Automation use restricted, becomes niche tool, quality poor
Expensive tools used by a few, shelfware, high license fees
S I M GROUP
© 2000, SIM Group Ltd.


GRESHAM
Test Environments -
the Brave New World
S I M
GROUP
© 2000, SIM Group Ltd.

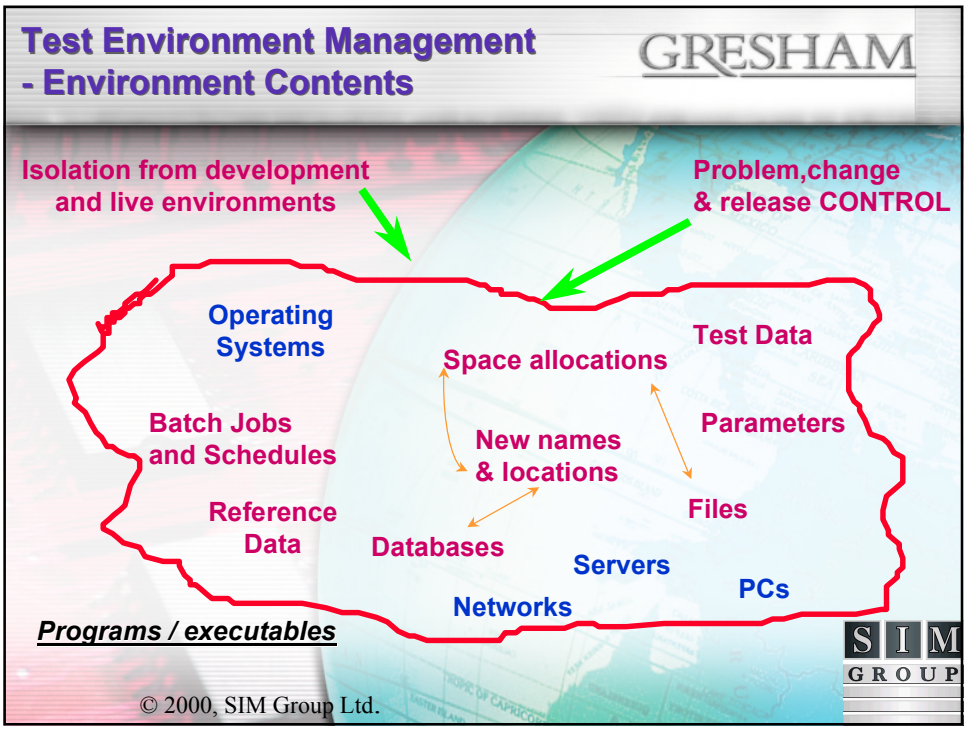

Test Environment Management
GRESHAM
Downsizing Pains while Growing
Non Mainframe platforms "Cheaper, Better, Faster"
Unix/PCs - Don't need Sys Progs, DBAs, Analysts, Programmers
Need other specialists - Unix, Network, PCs, DBAs!!!
Standards not required -
"It's a PC - standards don't apply"
"It's a Unix box - we don't need that stuff"
Project Lifecycles changed
Waterfall, CASE, RAD - Methodologies became fashionable!!!
All ignored testing and environments!!!!
S I M
GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
The Rise of the Internet and eCommerce
Early Websites became advertisement for Companies
Basic information and contacts details, not integrated
Normally single Server to keep isolated
Next level gave more information
Integrated to some systems - product and price info
Fill in order form and receive goods and invoice
Now we have eCommerce...
Fully integrated systems with up to date information
Credit card transaction, credit checking facilities
Multi-Server system, 24 hour, 7 days a week availability
© 2000, SIM Group Ltd.
S I M
GROUP

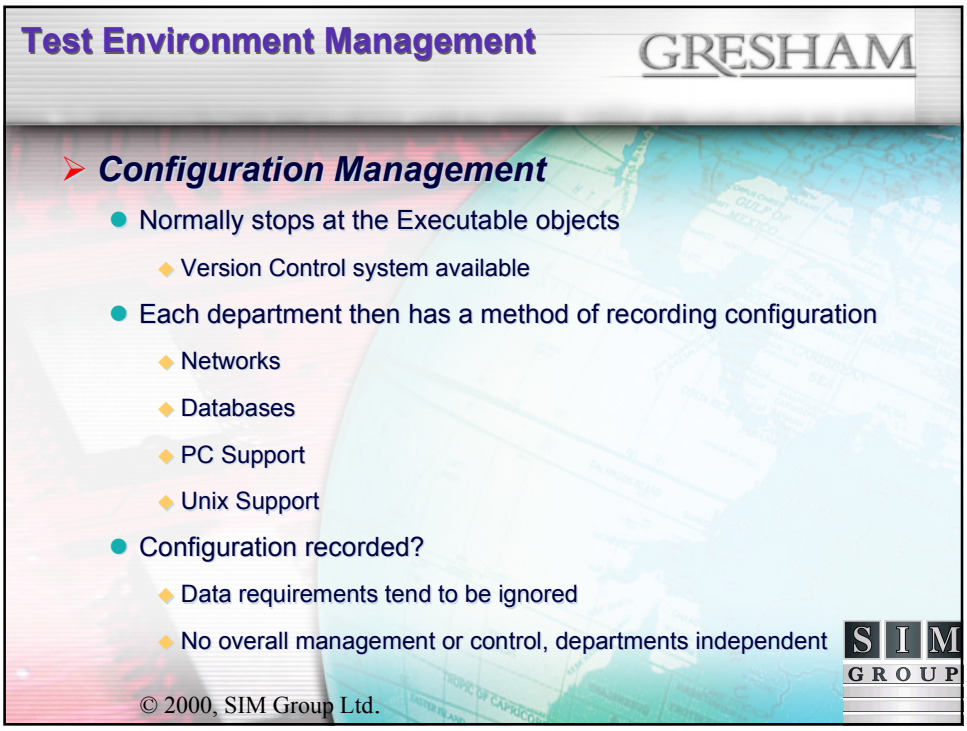

Test Environment Management
GRESHAM
Test Environments during this time...
Easy in early days…
Simple Server, run the tests using the limited data required
Security, Performance and Availability not seen as issues
Next generation
Security becomes the issue, environments difficult for testing
Performance testing raised in profile, again difficult to test
eCommerce requirements...
Environment with internal and external interfaces
Multi-Servers required, ability to test as Developer and Client
Security and Performance test capability a must
© 2000, SIM Group Ltd.
S I M
GROUP

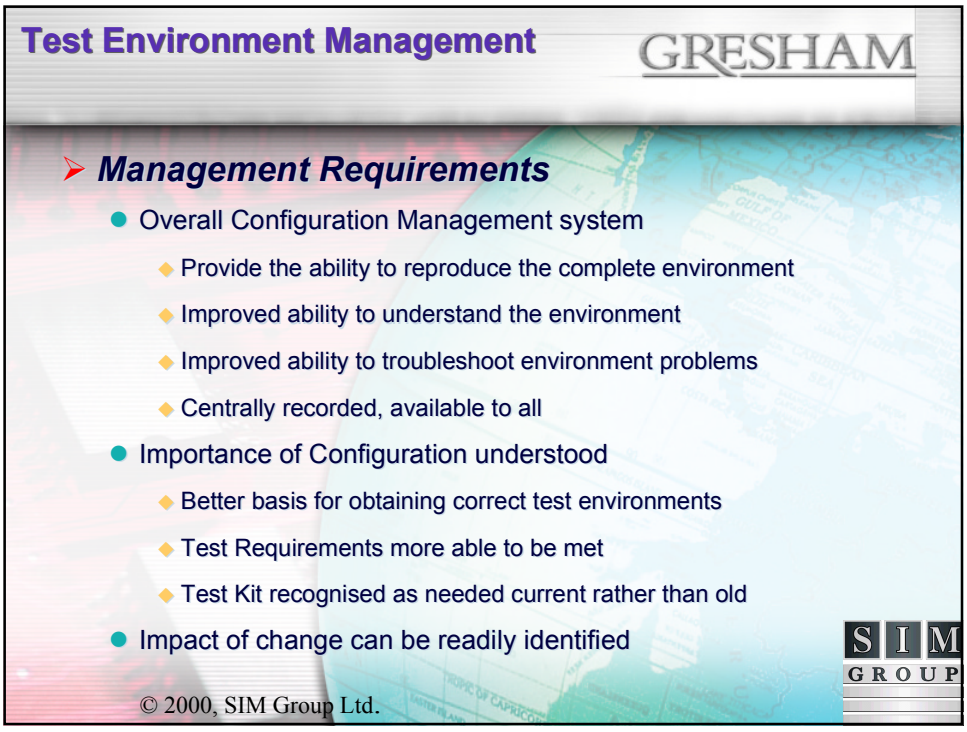

Test Environment Management
GRESHAM
The Future (is already here!!!)
WAP Technology
Systems already developed
Test Environment is a mobile phone!!
Palmtops
Requirement for more systems to interface
Web enabled and Physical linkages required
Test environment is a hand held box!!
Next hardware technologies - watches, wrist commands etc..
Test environment management principles still apply
The same mistakes will be made!!
© 2000, SIM Group Ltd.
S I M
GROUP


GRESHAM
Test Environments -
Configuration Management
Needs and Wants
© 2000, SIM Group Ltd.
S I M
GROUP


Test Environment Management
- Environment Contents
GRESHAM
Isolation from development
and live environments
Problem,change
& release CONTROL
Operating
Systems
Space allocations
Test Data
Batch Jobs
and Schedules
New names
& locations
Parameters
Reference
Data
Databases
Files
Servers
PCs
Networks
Programs / executables
SIM
GROUP
© 2000, SIM Group Ltd.

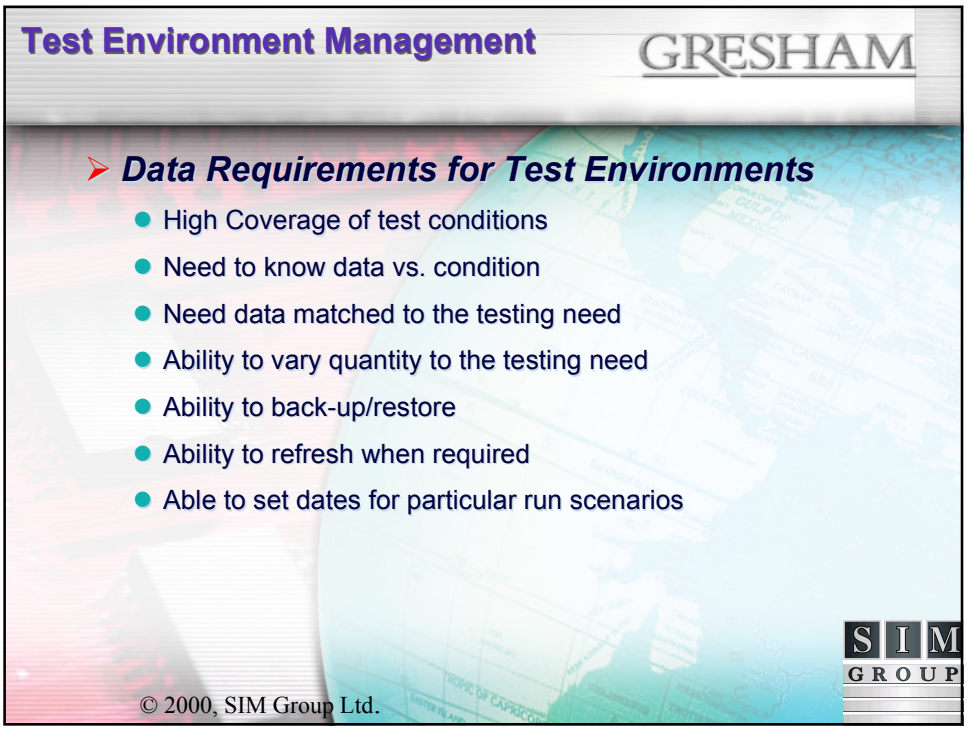

Test Environment Management
GRESHAM
Test Environment Creation
Much more technical exercise on today's Client/Server systems
Many more factors to control, configure and record
Setting up of Production environment copy not a necessity
Number of Servers reduced, Hard drive capacity reduced
Limited PCs available, "standard" PCs or found in corner
No "clean" environment with Production desktop configuration
Performance testing always an issue in today's systems
Configuration recorded?
Not always, if an element fails, could be no method of reproduction
Management systems not seen as a necessity
SIM
GROUP
© 2000, SIM Group Ltd.

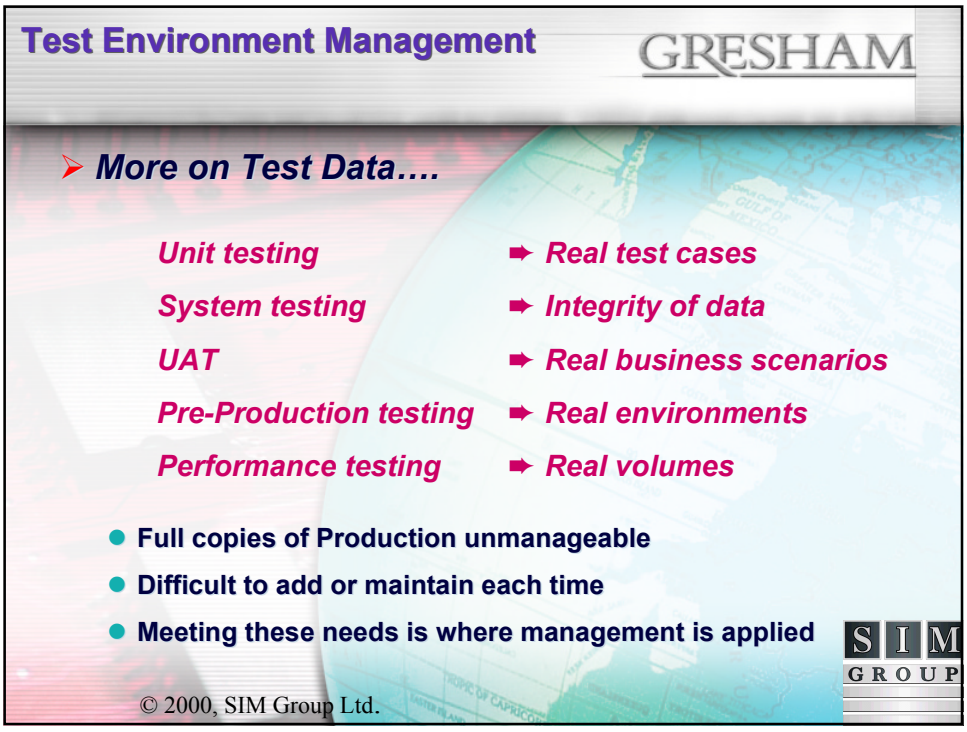

Test Environment Management
GRESHAM
Configuration Management
Normally stops at the Executable objects
Version Control system available
Each department then has a method of recording configuration
Networks
Databases
PC Support
Unix Support
Configuration recorded?
Data requirements tend to be ignored
No overall management or control, departments independent
© 2000, SIM Group Ltd.
S I M
GROUP


Test Environment Management
GRESHAM
Management Requirements
Overall Configuration Management system
Provide the ability to reproduce the complete environment
Improved ability to understand the environment
Improved ability to troubleshoot environment problems
Centrally recorded, available to all
Importance of Configuration understood
Better basis for obtaining correct test environments
Test Requirements more able to be met
Test Kit recognised as needed current rather than old
Impact of change can be readily identified
© 2000, SIM Group Ltd.
S I M
GROUP

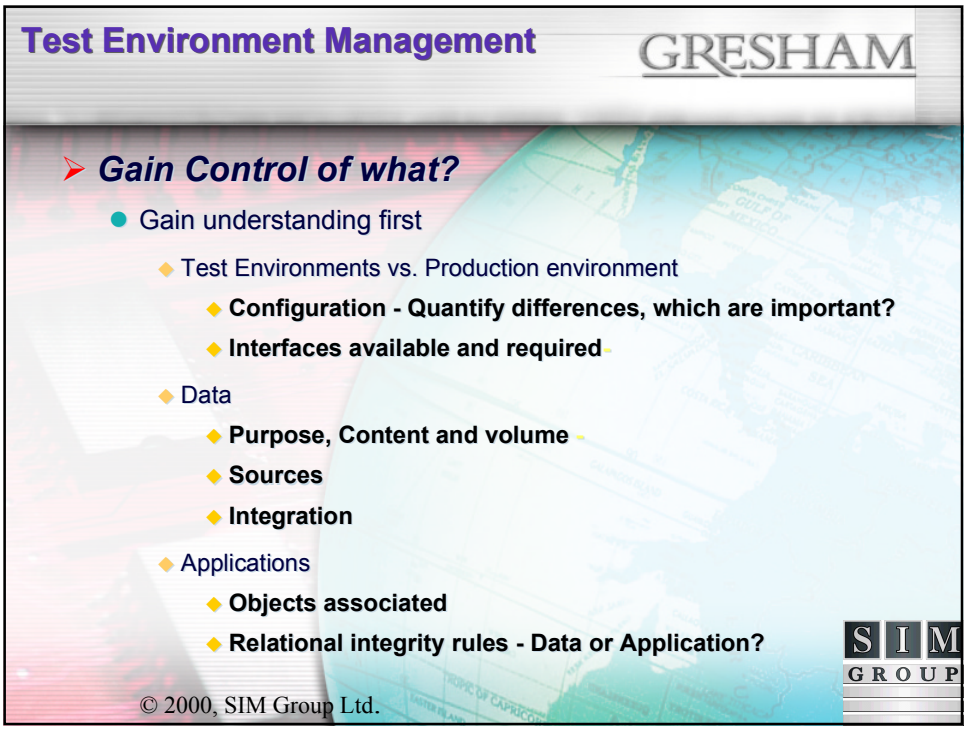

GRESHAM
Test Environments -
Data Requirements
S I M
GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Test Environment Data Requirements
High Coverage Test data
Base Test Data
Specific Test Conditions
Manageable Volumes of data
Responsive to project testing time frames
Able to support each level of testing
Ability to back-up/restore
Data must have integrity
Internal and External data interfaces
S I M
GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Data Requirements for Test Environments
High Coverage of test conditions
Need to know data vs. condition
Need data matched to the testing need
Ability to vary quantity to the testing need
Ability to back-up/restore
Ability to refresh when required
Able to set dates for particular run scenarios
S I M GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
More on Test Data….
Unit testing
System testing
UAT
Pre-Production testing
Performance testing
Real test cases
Integrity of data
Real business scenarios
Real environments
Real volumes
Full copies of Production unmanageable
Difficult to add or maintain each time
Meeting these needs is where management is applied
S I M GROUP
© 2000, SIM Group Ltd.


GRESHAM
Test Environment Management -
Gaining Control
S I M
GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Gain Control of what?
Gain understanding first
Test Environments vs. Production environment
Configuration - Quantify differences, which are important?
Interfaces available and required
Data
Purpose, Content and volume
Sources
Integration
Applications
Objects associated
Relational integrity rules - Data or Application?
S I M
GROUP
© 2000, SIM Group Ltd.


Test Environment Management
GRESHAM
Gaining Control
Determine boundaries of applications
Determine the data flow through applications
Understand how the application works
DOCUMENT ALL YOU HAVE FOUND!!!!
S I M
GROUP
© 2000, SIM Group Ltd.


GRESHAM
Test Environment Management - Implementation
S I M
GROUP
© 2000, SIM Group Ltd.

Test Environment Management

GRESHAM

- **So what is missing?**
  - Test Environment Management is a new role, part of test culture
  - The culture should ensure the environments are
    - Refreshable
    - Maintainable
    - Up to date
    - Flexible
    - Integrated
  - Data requirements must be able to be met
  - Configuration management is a must for all objects
  - **Project Team time on environments is too costly**

SIM GROUP

Test Environment Management

GRESHAM

- **First Steps, Environment basics**
  - Ensure all databases and files are latest configuration
  - Remove all data that is superfluous
  - Refresh Reference and Static data
  - Add interfaces required
  - Test the environment, back-up/restore procedures
  - Automate the processes for the above wherever possible
  - Seek a Project requiring an environment
  - Source the required data for the testing needs
  - Deliver the environment
  - This becomes your test for best process

SIM GROUP

Test Environment Management

GRESHAM

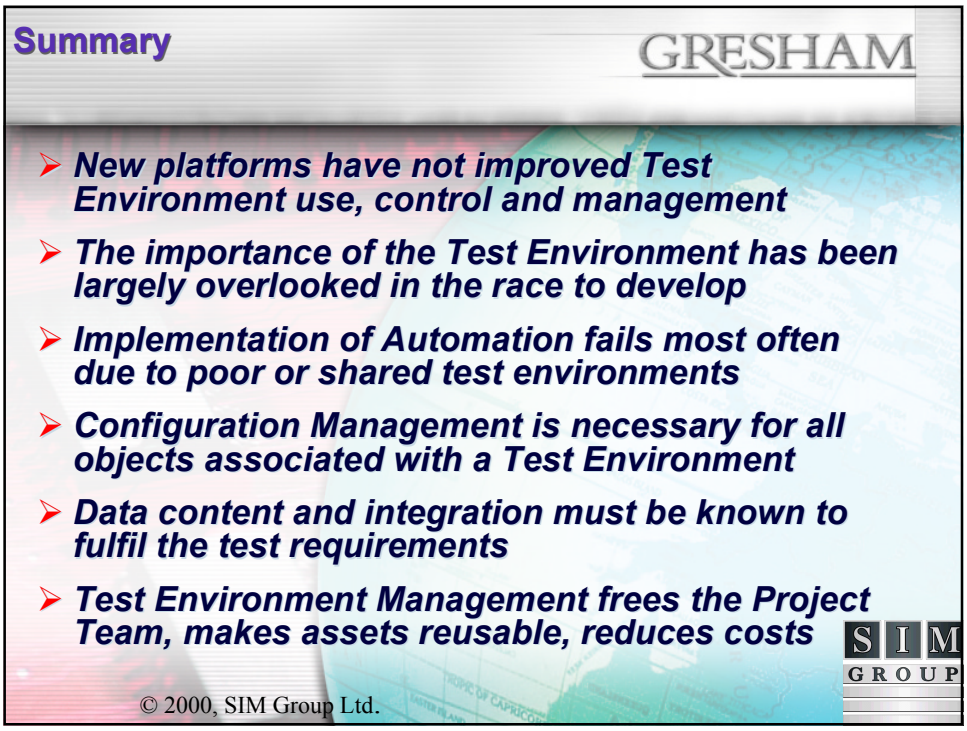
- ***Implement Best Process***
  - For Data Refresh and Maintenance
  - For Data Identification for Test Requirements
  - To build the test environment
  - To provide data volume scalability
  - For Environment Maintenance
  - To provide estimates, and fit into Project Lifecycle
- ***Automate Processes wherever possible***
  - Provides an easy method to build, populate and refresh
  - Gives greater control to manage environments

SIM GROUP

GRESHAM

Summary

SIM GROUP


Summary
GRESHAM
New platforms have not improved Test Environment use, control and management
The importance of the Test Environment has been largely overlooked in the race to develop
Implementation of Automation fails most often due to poor or shared test environments
Configuration Management is necessary for all objects associated with a Test Environment
Data content and integration must be known to fulfil the test requirements
Test Environment Management frees the Project Team, makes assets reusable, reduces costs
S I M
GROUP
© 2000, SIM Group Ltd.

QWE2000 Session 4I

Mr. Adrian Cowderoy
(ProfessionalSpirit Limited)

Complex WebSites Cost More to Maintain- Measure the Complexity of Content

Key Points

- Maintenance cost reduction
- Development Planning
- Change-rate control

Presentation Abstract

The proposed paper addresses the use of metrics and measures that describe the complexity and size of the different types of content within web-sites, for web-sites that are content rich.

These measures serve short-term commercial needs, such as:

- Maintenance cost reduction. Complexity measures provide indications of which digital objects are likely to be difficult to maintain. (The cost of maintenance for websites is typically 50-200% of the original development cost, and can be many times the cost of the original project.) Often high-risk objects can be designed differently, or documented better.
- Development planning. Simple size measures are useful for allocating work between the team. Complexity measures (and technical quality requirements) indicate the level of skills, testing and quality improvement methods that are needed.
- Change-rate control. Externally visible size and complexity measures are needed for defining the size of the product, so that subsequent major change-requests are charged separately.

Measures are also useful for giving crude indications of cost (such as via analogy-based estimation) and for simple benchmarks (such as to assess whether there is a significant difference resulting from the introduction of a new tool). However there is so much volatility in the development process, and so many uncontrolled influencing factors, that estimation and benchmarking are unreliable processes.

Web-sites consist of both content and functionality. The quantity and complexity of the content is typically the main cost driver in many software projects. Software is generated automatically from HTML editors, graphics tools and movie/animation

editors. Further (relatively simple) code may be added by staff who are not professional programmers. The effort to assemble the code is small, but the effort for testing and correction can be considerable. (A persistent problem with these tools is that the reliability and performance of their embedded code, such as in browser plug-in's, constrains the reliability and performance of the final product.)

Some web-sites involve also major software components - e.g. e-commerce sites. Such projects tend to perform overlapping projects, one for the software component and the other for the content, followed by a period of system integration. There are significant differences in the software and content development processes and skills. This paper addresses content development - the use of size and complexity for software is covered by the software industry.

The paper introduces different sets of size measures for each of the main concurrent activities within a web-development project. Table 1 below gives some examples of complexity and size measures for different types of digital object.

About the Speaker

Adrian Cowderoy is Managing Director of the Multimedia House of Quality Limited, a company which he established to promote quality-improvement methods for the production of websites and multimedia.

Mr Cowderoy was the General chair of ESCOM-SCOPE-99 and ESCOM-ENCRESS-98 conferences, and was Program chair for ESCOM 96 and 97 (The European Software Control and Metrics conference promotes leading-edge developments in industry and research, worldwide û see www.escom.co.uk). He is the METRICS-ESCOM Coordinator for IEEE METRICS 2001 and was on the Program committee of Metrics 98 and 99, European Quality Week 99 and COCOMO/SCM 96-99. In 1998 he was acting Conference Chair of the Electronics and Visual Arts conference in Gifu, Japan. He is a registered expert to the European Commission DGXIII.

He has provided consultancy and industrial training courses on quality management, risk management, and cost estimation to the aerospace and medical industries in the UK, Germany and Italy since 1995. He also lectures at Middlesex University (www.mdx.ac.uk) on e-commerce project management and managing Internet start-up's, and at City University, London (www.city.ac.uk), on project management for systems development.

Mr Cowderoy was project manager and technical director of MultiSpace, a 14-month million-dollar initiative sponsored by the European Commission in which 12 European organizations explored the potential to apply quality-improvement methods to multimedia and website development projects. (See www.mmhq.co.uk/multispace and www.cordis.lu/esprit.)

He was a Research fellow at City University from 1990-1998, and a Research Associate at Imperial College from 1986-1989. He was also a quality consultant and software developer at International Computers Limited, UK, from 1980-1985, where

he worked on operating and networking systems for mainframes and distributed systems.

His academic qualifications include an MSc in Management Science from Imperial College, University of London in 1986, and is a member of the Association of MBA's. He received a BSc in Physics with Engineering from Queen Mary College, University of London, in 1979.

Mr. Cowderoy has published and presented extensively on multimedia quality and software cost estimation. He was joint editor of Project Control for 2000 and Beyond (Elsevier, 1998), Project Control for Software Quality (Elsevier, 1999), and Project Control: The Human Factor (Elsevier, 2000).

PROFESSIONAL *Spirit*

Complex websites cost more to maintain

Adrian Cowderoy

Presented at Quality Week Europe, Brussels, 2000

Background

- **Web-sites**
  - functionality (e.g. Java, Lingo, etc)
  - content/assets (e.g. pictures, text, movies)
  - mixed elements (from application generators)
- **History**
  - Research: MultiSpace project (EP23066)
  - Development: MMHQ
  - Community: at ProfessionalSpirit
- **Influences**
  - Software Metrics Fenton & Pfleeger
  - GQM Basili & Rombach
  - FP's Albrecht & Gaffney

PROFESSIONAL *Spirit*

Motivation

Use of size metrics for Software

- for cost estimation
- for benchmarking
- for improving designs

… and for Content

- some different problems to solve

"Complex websites cost more to maintain"
QWE2000, page 3

PROFESSIONAL *Spirit*

Motivation for web-sizing

- **Identify high-risk components**
  - complex structures, extreme size, unusual combinations
- **Costing project changes**
  - charge-rate for changes, subject to constraints
- **Identify poor usability**
  - indicators of poor operability

Also

- task scheduling
- analogy-based estimates
- evaluation of new tools
- research

"Complex websites cost more to maintain"
QWE2000, page 4

PROFESSIONAL *Spirit*

Asset types

- **Logical structure**
- **Formatted text and hypertext**
- **Data content within databases**
- **Images**
  - including bitmaps, vectors, rendered 3D
- **Audio**
- **Interactive Movies**
  - including animation, interactivity

Also Interactive 3D, Motion, Smell, Physiological

"Complex websites cost more to maintain" QWE2000, page 5

PROFESSIONAL *Spirit*

Types of measure

- **Primary size measure**
  - main indicator of cost
- **Complexity of an asset or asset component**
  - indicates maintenance difficulty or increased development cost
- **Cohesive complexity**
  - indicates interaction between assets
- **Extent of interactivity**
  - functional component within the asset

+ Classification schemes that allow distinctions

"Complex websites cost more to maintain" QWE2000, page 6

PROFESSIONAL *Spirit*

Measures

- **Quality of Metrics**
  - relevance, consistency, precision, learnability, cost
- **Aggregation**
  - total size for identical elements
  - need warning when aggregation is invalid (e.g. from different complexity, different classes)
- **Derived measures**
  - combine 2 or more measures into a single measure
  - high utility, but very difficult to achieve reliably
    - example on next page

"Complex websites cost more to maintain" QWE2000, page 7

PROFESSIONAL *Spirit*

Example, for website structure

"If more microsites than pages, then technical integration may be difficult"

"If there is high density of external hyperlinks, then maintenance costs will be higher"

"If long sentence with long words (Fog Index) then rewriting sentence"

"If big image sizes with multiple layers then compression is likely to be inefficient"

"If library size is much larger than sprite count, then exceptional learning and maintenance"

"Complex websites cost more to maintain" QWE2000, page 8

PROFESSIONAL *Spirit*

Choosing what to measure

- **Choose what to improve**
- **Choose what to measure**
- **Choose how much to measure**
- **Choose when to measure**

"Complex websites cost more to maintain" QWE2000, page 9

PROFESSIONAL *Spirit*

Choose WHAT to improve

Identify complex objects - they require more testing, and they can be difficult to maintain and extend.
Identify exceptionally large objects - check to see the technical problems that may occur, plan to how to allocate the staff, and ensure there is sufficient design and maintenance documentation, and the files are properly catalogued.
Identify expensive components - measure their costs accurately, identify the risks of changed needs, identify how late the purchase can be made, and identify the copyright and legal issues.

"Complex websites cost more to maintain" QWE2000, page 10

PROFESSIONAL *Spirit*

Example for "graphics design"

- **Exceptional size comes from an exceptional number of *delivered_images* and *image_composites*, especially when there are constraints on *browser_compatibility*, *language_count* and *rigour*.**
- **Testing effort is increased by demands on *browser_compatibility*, *language_count* and *rigour*.**
- **Problems with acceptance of the final product may occur if there is high *rigour* and the *source_of_image* is outside the developers control.**
- **Maintenance is effected by the number of *image_composites* compared to the *image_size*.**

"Complex websites cost more to maintain" QWE2000, page 11

PROFESSIONAL *Spirit*

Choose how much to measure

- **Risk to quality**
- **Risk to schedule**
- **Process stability**
- **Technology stability**
- **Personal motivation**

"Complex websites cost more to maintain" QWE2000, page 12

PROFESSIONAL *Spirit*

Choose when to measure

- **Initial target**
- **Design completion**
- **First demonstration prototype**
- **Asset delivery**
- **Project completion**

"Complex websites cost more to maintain"
QWE2000, page 13

PROFESSIONAL *Spirit*

Summary

- **Content size is measurable**
- **Many complexity and structure measures**
- **Measurement is feasible**
- **Status**
  - enough exists for practical uses
  - opportunities for research, modelling and tools

"Complex websites cost more to maintain"
QWE2000, page 14

PROFESSIONAL *Spirit*

Web-activity

www.mmhq.co.uk/my-complexity

- size and complexity measures
- what to measure for specific
- examples

- contact: adrian.cowderoy@professionalspirit.com

Related topics

www.mmhq.co.uk/my-quality … web quality

www.mmhq.co.uk/multispace … project results

www.emmus.org … web usability

"Complex websites cost more to maintain" QWE2000, page 15

Complex website cost more to maintain

Adrian Cowderoy

PROFESSIONAL *Spirit*

Web-site content can be characterised in terms of size, element complexity, coherence and exceptions. Appropriate measures can be defined for all the common types of digital asset. These measures are available for use throughout the lifecycle.

A measurement plan for projects can be derived from a variant of GQM, the information that can meaningfully be provided by different activities, and the extent to the tolerance of the key development staff.

The use of complexity and size for improving website costs and quality has similarities to that for software development, however the detail and emphasis differs at almost every point.

1. Introduction

In developing the website and software for ProfessionalSpirit, we recognised from the start the importance of controlling project costs for the content-development activities as well as the software. To achieve this we use measures of size and complexity, adopting principles developed in the software industry by Fenton (for software metrics definition) and by Basili and Rombach (for Goal Question Metrics definition) [1][2]. The work also builds on the quality-oriented framework developed by MultiSpace (project EP23066 in the EU ESPRIT programme) [3]. Lists and supporting information can be found at the MMHQ website, http://www.mmhq.co.uk/my-complexity/

The paper begins by addressing the business objectives for size and complexity measures. The categorisation and measurement processes are then described. This is followed by a procedure for developing a measurement plan. Three case studies are presented. Finally there is a discussion, in comparison to the example set by the software industry.

Table 1: website terminology used in this paper

Content – all the elements of the website which contain no functionality, as in simple HTML pages. JavaScript, Lingo, etc, add functionality to page and movie content.
Assets –images, narrative, movies, etc, which collectively comprise the content.
Storyboard – the sequence of narrative and interactive events within each scene of a movie/animation, defined against the frame number.
Asset-component – a subdivision of an asset that is only accessible to its developer.
Lifecycle – website development involves parallel development of different types of asset, often with prototyping, and concluding with overall integration.

2. The business objectives

Basili and Rombach recommend that the effort to regularly collect measures of size and complexity must be justified by business questions, relating to business goals. For website production, our experience is that there are three major issues that effect most projects, and several secondary ones.

2.1. Identify high-risk components

The creation of a web-site can consume enormous resources from companies and organisations. Corrective and adaptive maintenance costs at least a third of the lifecycle costs, and often two-thirds. For continually changing websites the cost is even higher.

A complex website structure can increase maintenance costs considerably. Consequently every area of complexity in a website is a potential project risk. This paper presents measures that can be used to highlight unusual complexity in a digital asset. Such "high-risk" assets tend to benefit from careful monitoring, the assignment of skilled staff for development, and the use of more detailed testing. It may even be appropriate to rebuild the asset using less complexity.

Complexity also exists in the software component of websites. For e-commerce websites the costs and complexities can be considerable, but for most websites software represents only 10% of the project costs. This paper focuses on content, and the limited amounts of functionality resulting from code generation (as in JavaScript for rollovers) and from the relatively simple programming functions included in animation and movie applications (such Macromedia Flash and Director). Software complexity is dealt with elsewhere.

2.2. Costing project changes

The development of any website typically involves several iterations in which styles, graphics and content change. The cost for a limited amount of quality-enhancement is typically included in the budget, but to keep the project cost under control, the contract usually fixes the size (as illustrated in Table 2). Clients who later request major changes and extensions must re-negotiate the contract. Not only does this add further delay (in addition to that from the extensions), but it puts the client in a very weak position contractually.

A solution is to specify in the original contract a charge-rate for changes, subject to specific conditions being met. To achieve this effectively, suitable measures are needed for size and for each of the specific conditions. Table 3 gives an example.

Table 2: example of a traditional method of fixing the size of a web-site

Type	**Measure and specification**
Static pages	• Number of pages. • Specified nature of each page (i.e. type of assets it contains). • Structure of website.
Movies and sound	• Total duration. • Nature of each movie/sound element.

	• Specified nature of the required interactivity.

Table 3: basis for contract for late changes to the static pages in a website

Charge rates	**Conditions**
€ /page (for new page or respecified_page) + € for disruption cost	• Request made after prototype #1. • Extension to project deadline of twice the delay currently implied by the product size and project duration. Also a 1-week delay for reviewing, plus delays from re-negotiating the contract. • No change to quality requirements. • Complexity of each element is similar to before. • Costs of further bought-in assets are covered by the client. • The external risks to the project are not increased by the change.

2.3. Identify poor usability

Surveys conducted by the MultiSpace project (two using questionnaires and a third via interviews) revealed that almost every designer and project manager in multimedia and web production had usability as a high concern. These concerns can be typified by the questions:

- Is it difficult to get started?
- Is the interface quick and efficient?
- Is it difficult to get lost?
- Have ugly features been avoided?

Size and complexity measures can be used to indicate poor usability. These are only symptoms of poor usability, not proof, but they have the benefit of being available early in the development process, when it is still relatively easy to change the product.

2.4. Secondary business objectives

In addition to the three business objectives above, there are other uses for size and complexity measures, as shown in the bullet list below. If the information is currently available, these secondary objectives can be satisfied, however it is seldom possible to justify the collection of further data based only on the secondary business objectives.

- *Task scheduling.* Simple size measures provide indications of the relative length of each of the development tasks within the content production. Complexity measures indicate the skill levels required to build the digital assets. Increases in complexity or size compared to the original outlines, indicates a risk-to-schedule.
- *Analogy-based estimates.* Costing of website development effort can be made against an analogous project, with the differences in size treated as a proportional increase in cost. Differences in complexity indicate that the estimate may be wrong. (In some cases it may also be possible to estimate the cost-impact of a change in complexity.)

- *Evaluation of new tools.* New graphics and authoring tools often allow considerable increases in the complexity that can be handled by the same staff within a similar timescale. Some tools provide a considerable productivity improvement for routine adjustments and conversions. Such performance improvements can be measured. Performance changes are also important for tools that involve rendering animations, where exhaustive computing power is used – the effect of differences in complexity can lead to designs that reduce the computing requirements.
- *Research.* Size and complexity measures can contribute to our understanding of the dynamics of website development. Potentially tasks that have repeatable processes could be supported by locally-calibrated effort estimation models akin to those used in the construction and software industries [5].

In addition, each organisation may have further business goals that are not listed above.

3. Characterisation of the features

The different types of digital asset used within websites each requires different measures of size and complexity, as indicated in Table 4 below. The list of different asset-types used by MMHQ is adapted from the list published by the MultiSpace project [3].

Size and complexity measures can be defined for each of these asset-types. These measures describe internal characteristics of the components that provide indicators of some features of quality (as in the example of maintainability), but not are conclusive evidence. There are also many quality features that are not described by size and complexity [6].

Table 4: different types of asset used in websites

Asset-type	**Definition**
Web-site structure	Logical structures applied to the website. (Websites can be supported by multiple hierarchies, linear storyboards, and more.)
Hypertext	Formatted text and hyperlinks to other pages (in addition to the links implied by the web-site structure).
Database contents	Data content within databases (as in e-commerce websites). Also papers and presentations within libraries (as at the ESCOM website).
Images	Graphics design work. Includes bitmaps, vector graphics and rendered images. Also includes banners that have no storyboard.
Movies	Movies, from existing images, video, text and sound. A movie always has a linear storyboard, and may include extensive interactivity.
Audio	Sound tracks (including music, voice and effects)..
Interactive 3D *	Three-dimensional objects and the current generation of virtual worlds, with which the user can interact.
Motion *	Servo-assisted effects, such as in simulators and for robotic control. This includes remote control of house appliances via the web.

Smell *	Olfactory stimulation. Currently used in retail kiosks, but not yet for conventional websites.
Physiological *	Interaction between system and bodily functions. Currently used for diagnosis, but potentially also for stimulation.

* = asset-types currently not supported by MMHQ' s lists

The lists used at MMHQ support the most-common asset-types, however the lists of options are not exhaustive. Indeed, they are kept as short as possible for operational reasons – the technology is not stable (and this can effect the size measures), the willingness to use measures is limited, and usability of the service is improved by a short list.

Some assets are constructed from component elements that are similar in nature to the entire asset. The complexity of these asset-components may different substantially within an asset, and consequently size and complexity measures are required may best be applied to each asset-component, rather than the entire asset. For example, a movie may consist of several scenes, and a displayed web-page may be built automatically from several texts and images.

For each of the assets and their component parts, a distinction can be made between different features, each of which is used differently. Specifically:

- *Primary size measure* is the main indicator of development cost and (for bought-in assets) purchase price. The choice of primary size measure follows industry conventions.
- *Complexity of an asset or asset component* indicates maintenance difficulty or increased development cost. (A lack of complexity, can also be significant where this has resulted from a digital worker not maintaining individual editable layers in the image.) Some complexity measures are also alternative size measures, but their effect on cost overlaps considerably with the primary size measure.
- *Cohesive complexity* indicates the extent and nature of the interaction between the asset (or its component parts) with other assets, either locally or anywhere on the web.
- *Extent of interactivity* refers to the functional component within the asset. This may include both user-written software code and application-generated software. (If there is complex functionality, then the interactivity would normally be developed by professional programmers, and the size and complexity measured using software metrics.)
- *The exceptions* provide classification schemes that allow distinction between assets on the basis of quality features, origin of the assets, and the use of technology. (The classification schemes use ordinal and nominal scale metrics.)

4. Metrics and measures

A metric is needed for each size and complexity feature in order for it to be meaningfully used. An important constraint in choosing and defining these metrics was the need to support people with different backgrounds, many of whom are not engineers. This involved careful attention to choice of metrics, emphasis of the importance of counting rules, and the dangers associated with aggregation and derived measures.

4.1. Choice of metrics

The choice of whether a metric is effective was based on five criteria: relevance, consistency, precision, learnability and cost-of-use.

Counting rules contribute directly to the criteria of consistency. Common experience of using measures in the software industry is that many of the measures are "subjective" – i.e. the counting rules are vague, and so consistency is reduced. The software industry also encountered metrics that are so complex that people tend to ignore the counting rules and consequently they count inconsistently. Measures need to be chosen that are easy to learn and cheap to collect.

Potentially a size and complexity feature could be assessed using more than one metric, however it helps to improve usability of the strategy if there is only measure per feature.

4.2. Aggregation

Often the size of an asset is identical to the sum of the size of each of the its component parts. This is convenient for costing, because purchase and asset-rework costs are closely related to size. However there are important exceptions when the whole is different to the sum of its parts – this is caused by duplication, overlap and blank sections. Consequently each size measures needs a "health warning" of when it is valid to calculate a "total size".

Aggregation of complexity measures is only sometimes meaningful, and has to be assessed on a case-by-case basis. It is seldom meaningful to have a "total complexity", but forms of "average complexity" are usually feasible. Some nominal and integer scale measures of complexity have exist in reasonably large samples which appear to follow a standard distribution (bell-shaped), and can be described using a statistical mean and standard deviation. Most other measures can only be meaningfully reviewed in terms of modes (most common value), upper and lower quartiles (relative to the mode), and the outliers beyond these quartiles.

Aggregates of "exceptions" is only meaningful using derived measures, as below.

4.3. Derived measures

Derived measures combine two or more different measures to produce a single meaningful number. A limited number have been identified by MMHQ, but there is opportunity for much more research work in this area. Some examples are given in Table 5 below.

Table 5: examples of how derived measures can indicate problems

For a website structure, if the number of number of microsites is great compared to the number of pages, then project coordination and technical integration may be difficult.
If there is a very high number of external hyperlinks compared to the number of words, then maintenance costs will be higher.
Long sentences with long words are confusing – the Fog Index counts the total number of words in a sentence plus the number of words with 3 or more syllables.
Big image sizes with multiple layers tend to result in files that compress inefficiently, and consequently cause performance problems.
For an interactive movie, if the library size is huge compared with the number of layers (or sprites) then more exceptional effort is needed for studying and maintaining the library.

5. Choosing what to measure

The potential list of measures exceeds what can realistically be collected in any one project. Consequently a choice has to be made of what to measure, and what to ignore. The strategy outlined in the section was inspired the work of the AMI project for software metrics collections [7], but the details are different as a result of the nature of website production. (Website development involves the use of multi-disciplinary teams from non-engineering backgrounds, unstable technologies, loose quality requirements, and a limited historic perspective.) The preferred strategy has four steps, as in the subsections below.

5.1. Choose what to improve

In section 2 above, a set of standard business objectives were presented for collecting size and complexity measures. For each of these standard objectives there are various standard suggestions of how to monitor for problems, and make improvements. Table 6 provides an example. The project members must select which of these Improvement Suggestions is appropriate for their own project. They may add further suggested improvements.

Table 6: standard list of Improvement suggestions for the business objective of "identifying at-risk components"

Identify complex objects - • they require more testing, and • they can be difficult to maintain and extend.
Identify exceptionally large objects - • check to see the technical problems that may occur, • plan to how to allocate the staff, and

• ensure there is sufficient design and maintenance documentation, and the files are properly catalogued.
Identify expensive components - • measure their costs accurately, • identify the risks of changed needs, • identify how late the purchase can be made, and • identify the copyright and legal issues.

5.2. Choose *what* to measure

A match can be now be made of which measures might be useful in support of each activity. For each measure, an explanation is needed of **why** the measure is appropriate and **how** it should be interpreted. If there is overlap between this and the Improvement Suggestions, then it is a good candidate for inclusion.

Table 7 below presents an example (and in this case, identifying components that could be difficult to maintain or develop). Note how measures are used for as many purposes as possible so as to reduce the extent to which measurement is required.

Table 7: example of measures for two development activities, in response to the business goal of "identifying at-risk components"

Activity	**Explanation and measures**
Graphics design	• Exceptional size comes from an exceptional number of *delivered_images* and *image_composites*, especially when there are constraints on *browser_compatibility*, *language_count* and *rigour*. • Testing effort is increased by demands on *browser_compatibility*, *language_count* and *rigour*. • Problems with acceptance of the final product may occur if there is high *rigour* and the *source_of_image* is outside the developers control. • Maintenance is effected by the number of *image_composites* compared to the *image_size* (if there are too few, or poor documentation, then problems occur).
Interactive movie production	• Exceptional size comes from exceptional *duration* for each type of origin (*source_of_movie*). • Testing effort is increased if there is high *rigour* applied to many *movies*, and they involve large *libraries* or large *object_counts*. • Testing problems are created by a *movie_source* that does not match the required *rigour.* • Software testing and maintenance costs are increased by the number of *decision_points.*

	• Lack of raw computer processing power comes from high *image_size*, *frame_rate*, and *duration.*

Items in *italics* refer to measures

5.3. Choose *how much* to measure

The overall extent to which an organisation is prepared to use measurement is influenced by various factors, as listed below. For measurement-adverse projects, the list of measures identified via the previous two subsections has to be further reduced to a level that will easily be tolerated by staff. (If the quantity of measurement is too high, it will be collected badly.)

- *Risk-to-quality.* Projects with high risk-to-quality can benefit from more detailed monitoring analysis of at-risk components. This requires measurement.
- *Risk-to-schedule.* When projects have highly compressed timescales there is little motivation to perform tasks that are not directly productive.
- *Process stability.* Organisations that have frequently-repeated activities, such as image digitisation, benefit more than others from the use of measurement.
- *Technology stability.* Some technologies are evolving rapidly and complex issues become easy, and new tool features are introduced that create new types of complexity. This considerably reduces the benefits of measurement. (Currently encountered in 3D and VR.)
- *Personal motivation.* Every person involved in the collection, analysis and use of data has to see direct benefits to their own work responsibilities. Thus designers are interested in measures that contribute to problem-reduction, but are not interested in charge rates and staff planning.

5.4. Choosing *when* to measure

The size and complexity metrics are (usually) applicable throughout the lifecycle, however the measurements made for these metrics may change at each milestone. These changes can highlight risks. The changes are also useful for analysing completed projects to understand the development processes.

Suitable milestones for collecting metrics include those listed below:

- *Initial target,* measured from the outline scoping of the product.
- *Design completion,* measured from the storyboard outline or the sketched text or image.
- *First demonstration prototype,* measured following the first demonstrable version. (Such prototypes often miss key features which will be provided later.).
- *Asset delivery.* The completed asset delivered for integration with the software and other assets.
- *Project completion.* The modified asset after late changes are made (for whatever reason).

6. Three case studies

The reality of using measurement is often different to the intended strategies. Three contrasting case studies are given below.

6.1. Dynamic website maintenance

The objective was to define practices that could reduce maintenance problems. The use of measurement could be part of this, or used as an argument for other new practices.

Selected measures. The standard metrics that are used to indicate maintenance problems (as in Table 7 above). This includes the primary size metrics that are also useful for costing maintenance changes.

Human issues. The measurement of the data occurs at the initial scoping and upon asset delivery, but not at milestones between or after (because of the extra limited benefits). The collection activity depends on the enthusiasm of an individual, but work is progressing to make it institutionalised. As of writing, general plans are in place for the data analysis, but there is insufficient data to justify a full analysis.

6.2. Small web-site production

The objective was to help with costing and quality control for solo website developers who produce large numbers of small websites for small businesses and community groups.

Selected measures included primary size drivers (for project costing), usability-related metrics (for use as guidelines), and limited image metrics (to avoid project cost escalation).

Human issues. There is a tendency for measures to increase awareness of good practice, but not to be used except in response to immediate problems.

6.3. First use of a new development tool

The objective was to determine the effect of switching from animated-GIF' s generated using Macromedia™Fireworks 2 to those produced by Macromedia™Flash 5. These are different types of tool, with the Flash providing extensive animation but lacking the graphics control of Fireworks. However, provided the output is animated-GIF, the products are comparable.

Selected measures. Some graphics vector tools have limited animation, others the opposite. The selected measures represent both these extremes. They thus highlight the overlap (like image size and duration) and the differences (like animation tweening).

Observations. The two packages encourage people to develop different types of animated GIF' s, which confuses the comparison. (Fireworks favours dramatic images and text changed stepwise, while Flash favours smooth transitions of simple objects.) The measures suggest that Flash only offers better maintainability if the library facility is used extensively.

Human issues. The exercise was performed by a single person, It helped him recognise that Flash was powerful and exciting, but should not be over-used. The immediate benefit of the comparison was that the exercise improved the subject' s understanding of the potential of the new tool. The longer term benefit is an indication of development costs.

7. Discussion: a comparison with software

A comparison between website development and software development is useful at This subsection presents revealing the differences that result from different kinds of product and different working methodologies (e.g. multi-disciplinary instead of engineering-based).

In software, a common method of pricing product changes is to use function points. In website development, the nature of the "product" is evolving and it is not possible to define a universal and timeless measure of size. Instead, the problem of costing contract changes can be resolved by using limited measures and constraints. (There is an opportunity for research to test whether this would also be a better strategy for software development projects.)

The "engineering" community within the software industry advocates process maturity towards statistically controlled processes with continual optimisation – a transition that may take 5-10 years to achieve. In website development, short-term flexibility and use of new technologies results in much higher financial rewards than process improvement. This impacts on the use of measurement: software engineers regularly collect measurements as part of their work processes, but website developers will collect measurement only for specific projects or in support of regularly repeated (small) activities.

Website developers have an extensive range of size measures that are available early in the project, and can be easily collected. In contrast, software development relies mainly on lines of code (which is available very late in the project) or function points (which are expensive to collect, not universally applicable, and can be inconsistently measured.)

Overall, the opportunities for using measurement in website development may be ultimately be stronger than for measurement of software projects.

8. References

[1] Fenton, N.E., "Software Metrics: A Rigorous Approach", Chapman & Hall, 1991.

[2] Basili, V.R. and Rombach, H.D., The TAME project: towards improvement-oriented software environments. IEEE Transactions on Software Engineering, 14(6), 758-773.

[3] Cowderoy, A.J.C., Daily, K., (editors): The MultiSpace Framework, ESPRIT EP23066 D2.2P. (Available at http://www.mmhq.co.uk/multispace/)

[4] MMHQ' s Website Complexity and Size Measures. A regularly changing list found via http://www.mmhq.co.uk/my-complexity/

[5] Cowderoy, A.J.C., "Size and quality measures for multimedia and web-site production", 14th COCOMO/SCM Seminar, October 1999, Los Angeles.

[6] Cowderoy, A.J.C., Donaldson, A.J.M., Jenkins, J.O., "A metrics framework for multimedia creation", 5th IEEE Software Metrics Symposium, Maryland, 1998.

[7] Pulford, K., Kuntzmann-Combelles, A., Shirlaw, S., "A quantitative approach to Software Management. The *ami* Handbook." Addison-Wesley, London, 1995.

QWE2000 Session 4M

Mr. Andreas Birk & Wolfgang Mueller [Germany]
(Fraunhofer Institute)

"Systematic Improvement Management: A Method For Defining And Controlling Customized Improvement Programs"

Key Points

- Why software engineering requires continuous improvement
- How to customize software process improvement
- The benefits of knowledge management for SPI

Presentation Abstract

Improvement has become an established practice in software engineering since the late 1980Ëies. Many different improvement approaches have been developed. Examples are process assessments, software engineering measurement, and quality management systems. However, still too often improvement programs are cancelled or do not reach their initial goals. We argue that this is due to two main reasons:

(1) Most improvement approaches have been developed in isolation from each other. Each approach has its specific strengths and application prerequisites. Only very little guidance is available for integrating different improvement approaches so that they complement each other and leverage the effectiveness of improvement programs.

(2) Appropriate management support for improvement programs is widely lacking. As a consequence, improvement programs take too long and consume too many resources. So improvement initiatives might even lose the support of their sponsors and the software development teams.

Systematic improvement management can overcome these issues and ensure the success of improvement programs. Improvement management must take care that the applied improvement strategy is appropriate for the given goals and characteristics of the software organization. In addition, it must monitor and control the improvement program during its execution and finally demonstrate that the improvement goals have actually been attained.

The paper and the presentation will introduce an improvement method that places particular emphasis on systematic improvement management. It covers the following

elements:

* The identification and explicit definition of business-related improvement goals, current software development practices, and experience that will be relevant to the improvement program.

* The integration of arbitrary improvement methods (such as process assessments, measurement, organizational learning, etc.) into an improvement strategy that meets the goals and characteristics of the software organization.

* A collection of typical improvement strategies for different types of software organizations (e.g., large embedded systems manufacturers, small or mid-size enterprises, or large banks or insurance companies).

* The monitoring and control of improvement program execution, including the risk management for improvement programs.

* The evaluation of improvement results, including the identification of new improvement goals and possible follow-up activities.

* An organizational infrastructure for running improvement programs and support instruments for facilitating the improvement management tasks.

About the Speaker

Andreas Birk is a senior consultant at the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE) in Kaiserslautern, Germany. His work areas are software process improvement, knowledge management, and organisational learning.

Andreas Birk was workpackage leader in the European technology development and transfer projects PROFES (ESPRIT 23239) and PERFECT (ESPRIT 9090). He has developed and evaluated several improvement methods for different types of software organisations. As a consultant, he has been working with many European software companies in the build up and extension of their process improvement programmes.

Andreas Birk has received a masters degree in computer science and economics (Dipl.-Inform.) from the University of Kaiserslautern, Germany, in 1993. He is currently applying for his PhD. He is a member of the IEEE Computer Society, GI, and ACM.

Dr. Wolfgang Mueller is leading technology transfer projects at the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE) in Kaiserslautern, Germany. His work areas are experience-based improvement programs and systematic knowledge management in the area of software development. Since April 1998 he is group leader in the department "Systematic Learning and Improvement" at the Fraunhofer IESE. From 1990 to 1997 he worked as scientist at the Fraunhofer Institute for Production Systems and Design Technology (IPK) in Berlin, mainly in the area of business process redesign and development decentralized shop floor control

systems.

Wolfgang Mueller has received a Ph.D. in engineering (Dr.-Ing.) from the Technical University Berlin, Germany, in 1997. He is a member of the German Computer Society (GI).

Systematic Improvement Management

A Method for Defining and Controlling Customized Improvement Programs

Andreas Birk
Wolfgang Müller

Fraunhofer IESE
Institut Experimentelles Software Engineering

Fraunhofer IESE
Sauerwiesen 6
D - 67661 Kaiserslautern
Germany

IESE

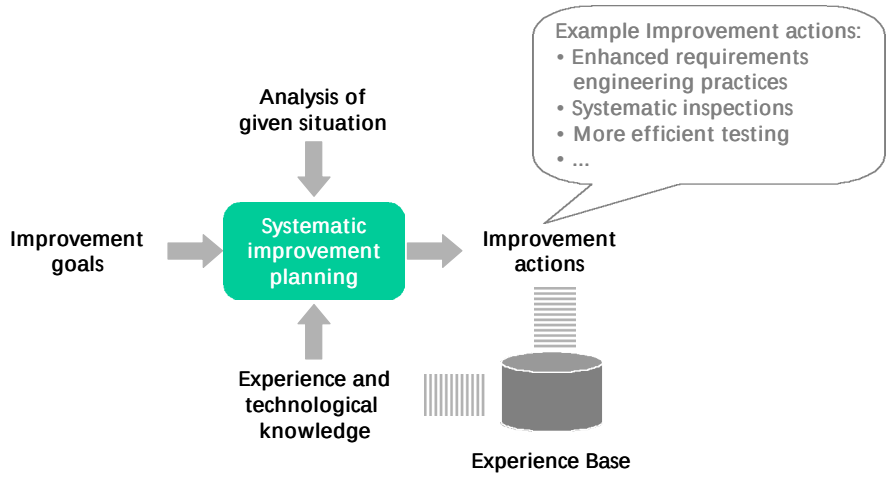
Motivation

- Improvement has become an established practice in software engineering
- Established improvement methods
  - Process assessments (CMM, ISO 15504, BOOTSTRAP, ...)
  - Process modelling and process guidance
  - Software and process measurement (GQM, PSM, ...)
  - Quality management systems / ISO 9000
  - ...
- However, still too many improvement programmes fail or perform badly
- Systematic improvement: Consolidate and advance improvement practices
  - Plan and manage improvement programmes systematically
  - Integrate improvement methods effectively
  - Tailor improvement strategies to the software organisation

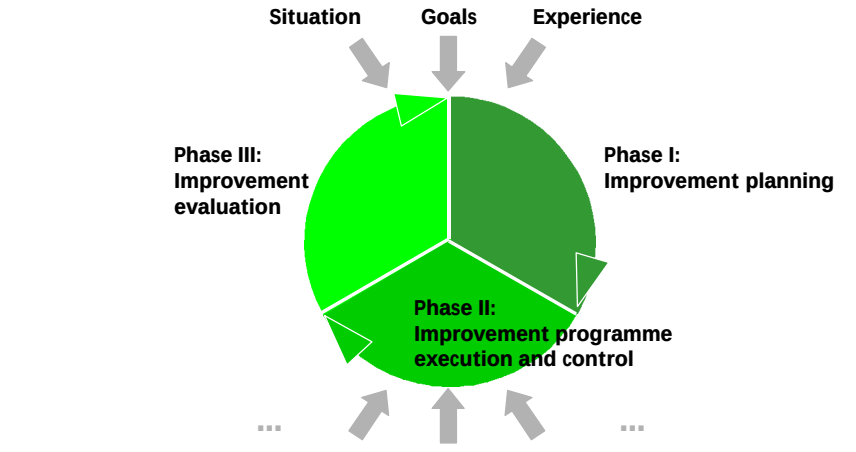
Principles of Systematic Improvement

- Improvement programmes must be driven by business-related goals
- Improvements must be attained rapidly and with little overhead effort
- Improvement strategies must be tailored to existing SE practices
- Improvement programmes must be planned and managed well
- The success of improvement programmes must be demonstrated
- Gradually, the software organisation capitalises on its own experience

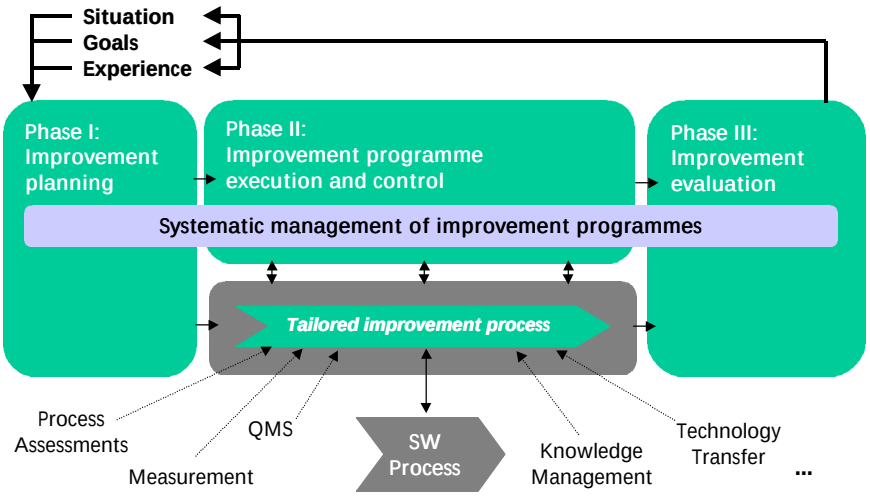
Systematic Improvement



Systematic Improvement



Strategic Improvement



Systematic Improvement Programmes

Phase I: Improvement planning

1. Analysis of the given situation
2. Identification and definition of improvement goals
3. Planning and adaptation of improvement strategy

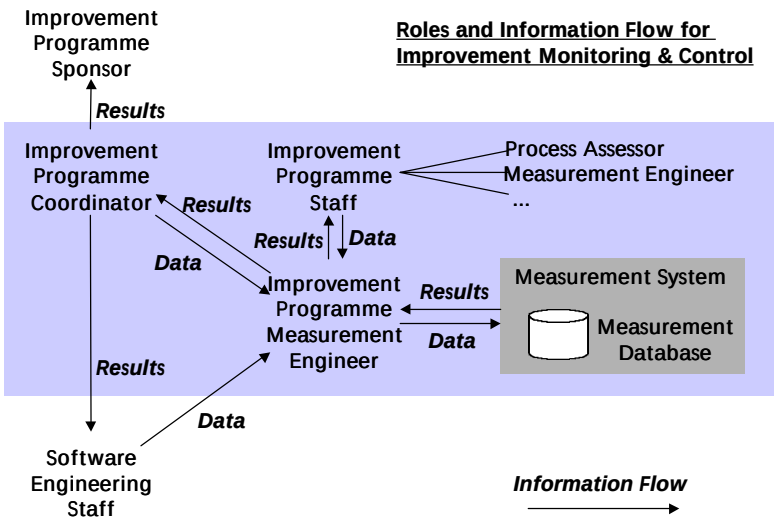
Phase II: Improvement programme execution and control

4. Execution of the improvement programme
5. Monitoring and control of the improvement programme

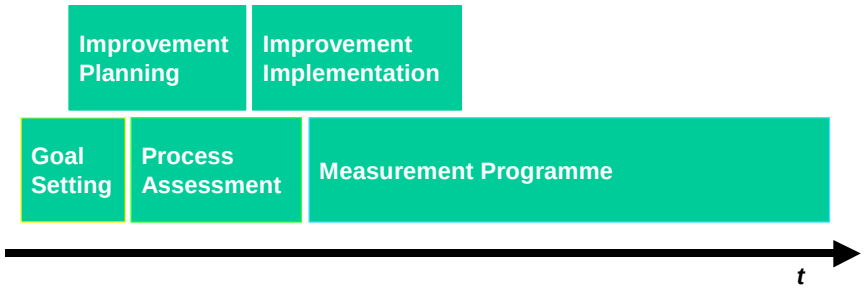
Phase III: Improvement evaluation

6. Analysis and evaluation of the improvement programme
7. Packaging and dissemination of the gained experience

Organisational Infrastructure for Improvement



Example of a Systematic Improvement Programme



Example of Improvement Programme Effort

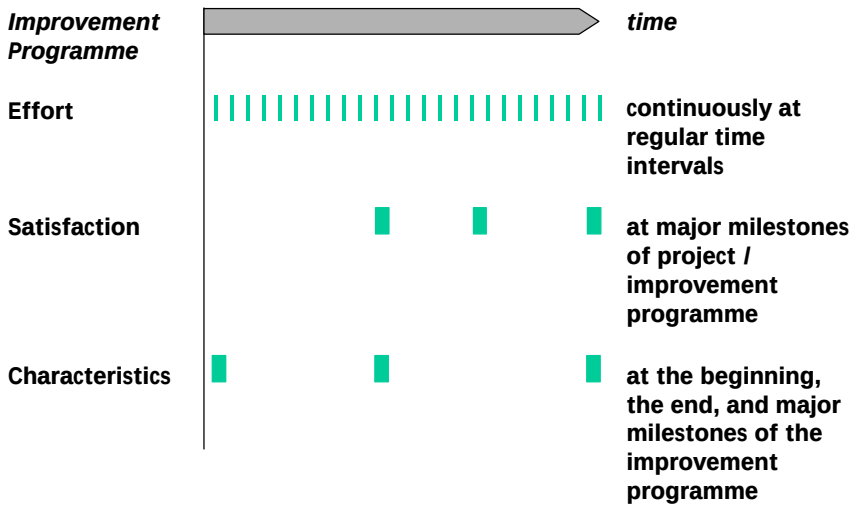
	Person Months	Calendar Weeks
Start up & goal setting	0.5	2
Process assessment	2.5	6
Measurement programme	2.5	40
Identification of PPDs	0.5	2
Improvement implementation	0.5	2
Total	6.5	52

Duration of one product improvement cycle:
Possibly less than one year

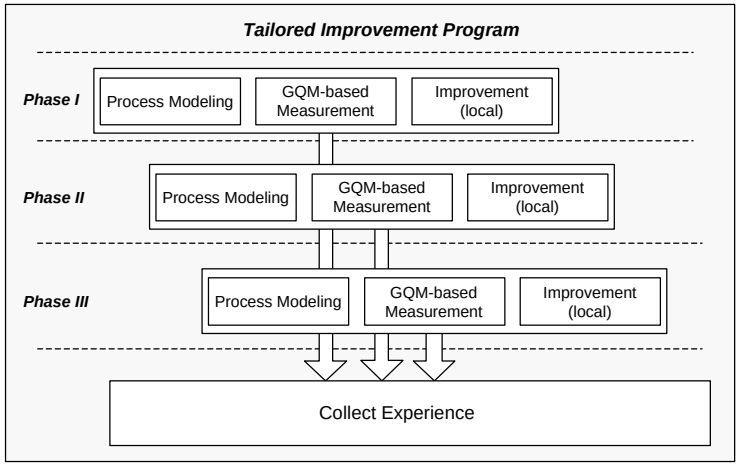
Three Core Measures for Improvement Programmes

Measurement is a prerequisite for monitoring, controlling, and evaluating improvement programmes.

When to measure and use which data?



Applications of Systematic Improvement



Applications of Systematic Improvement

Experiences from applying systematic improvement in the PROFES project:

- **Dräger Medical Technology** (Medical electronics)
- **Ericsson Finland** (Telecommunication)
- **Tokheim** (Systems and services for gas stations)

Attained product quality goals:

- Functionality
- Time-to-market
- Defect density
- Design quality
- Quality of the product architcture
- Requirements traceability
- ...

Process improvements:

- Process capability maturity improved from below level 2 to nearly level 3 within 18 months
- Attained the ISO 9001 criteria
- More effective design inspections
- Software measurement established
- Improved defect management
- ...

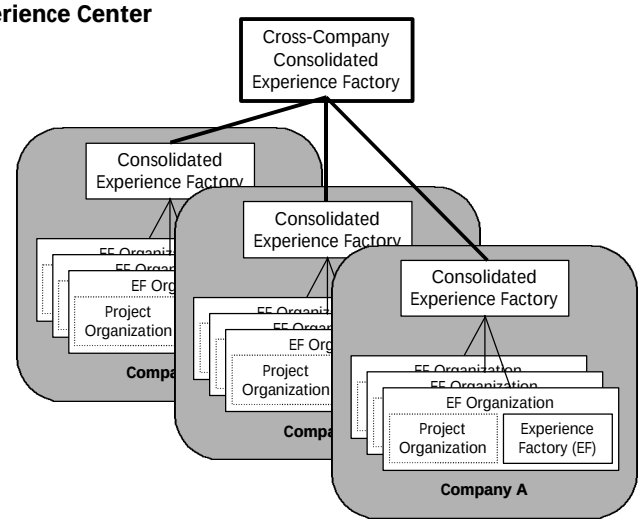
Improvements were attained fast and with little overhead effort.

SEC: Experience Exchange about Improvement

SEC™ = Software Experience Center

SEC Member Organizations:

- **ABB**
- **Boeing**
- **DaimlerChrysler**
- **Motorola**
- **Nokia**
- **Fraunhofer IESE**
- **Fraunhofer Center Maryland**



Summary

- A systematic improvement method is needed for the effective guidance of improvement programmes.
- Improvement processes must be tailored to the specific goals and characteristics of a software organisation.
- Knowledge mangement is important for long-term benefit from improvement programmes.
- Systematic management and control of improvement activities ensure the success of systematic improvement programmes.

Systematic Improvement Management: A Method for Defining and Controlling Customized Improvement Programs

Andreas Birk, Wolfgang Müller

Fraunhofer Institut für Experimentelles Software Engineering (IESE)
Sauerwiesen 6, D-67661 Kaiserslautern
{Andreas.Birk, Wolfgang.Mueller}@iese.fhg.de

Abstract

Improvement has become an established practice in software engineering since the late 1980'ies. However, still too often improvement programs are cancelled or do not reach their initial goals. In order to increase the success of improvement programs, they must be planned and controlled systematically by specialized management functions. This paper presents a method through which improvement programs can be customized to the specific goals and characteristics of a software organization. The method also covers the controlled execution of improvement programs and the evaluation of improvement success. Three case reports of different improvement strategies for different kinds of industrial software organizations illustrate the presented method.

Keywords: software engineering, process improvement, knowledge management

1 Systematic Improvement

Improvement has become an established practice in software engineering since the late 1980' ies. Many different improvement approaches have been developed. Examples are process assessments, software engineering measurement, and quality management systems.

However, still too often improvement programs are cancelled or do not reach their initial goals. We argue that this is due to two main reasons: First, most improvement approaches have been developed in isolation from each other. Each approach has its specific strengths and application prerequisites. Only very little guidance is available for integrating different improvement approaches so that they complement each other and leverage the effectiveness of improvement programs. Second, appropriate management support for improvement programs is widely lacking. As a consequence, improvement programs take too long and consume too many resources. So improvement initiatives might even lose the support of their sponsors and the software development teams.

Systematic improvement management can overcome these issues and ensure the success of improvement programs. Improvement management must take care that the applied improvement process is appropriate for the given goals and characteristics of the software organization. In addition, it must monitor and control the improvement program during its execution and finally demonstrate that the improvement goals have actually been attained.

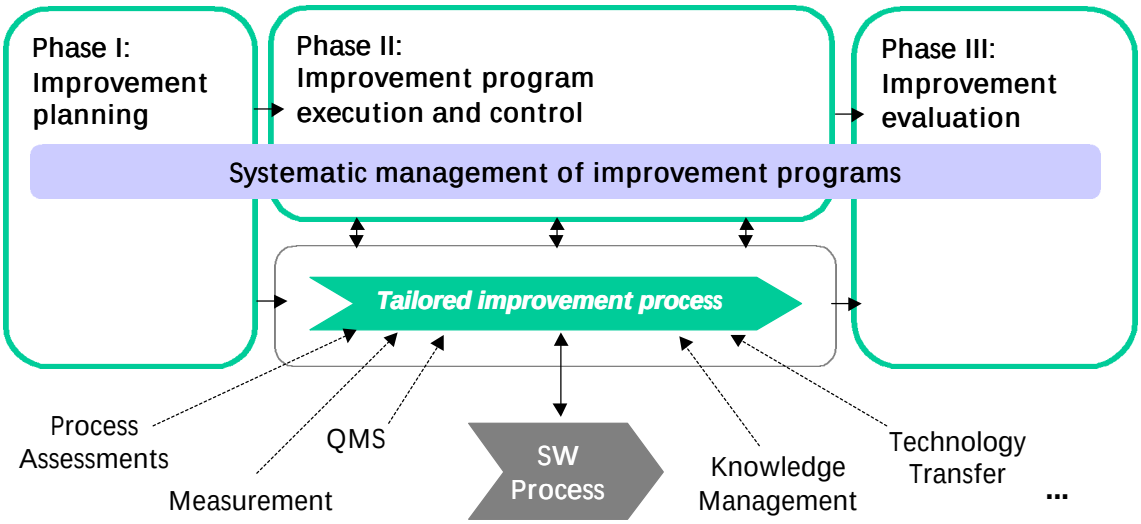


Figure 1: Elements of strategic improvement.

This paper introduces an improvement method that places particular emphasis on systematic improvement management and separates the management perspective from the technical aspects of improvement programs. A tailored improvement process must be defined that depends on the specific goals and characteristics of the software project and organization. It is built from improvement methods such as process assessments, measurement, and technology transfer. These basic elements of systematic improvement programs are introduced in the remainder of this section. Section 2 defines a method for defining and controlling customized improvement programs, which includes the managerial improvement tasks of systematic improvement. Section 3 reports two scenarios of tailored improvement processes from two European industrial software organizations. The important role of knowledge management for systematic improvement is discussed in Section 4.

1.1 The Elements of Systematic Improvement

The key elements of systematic improvement programs are shown in Figure 1. The systematic management of an improvement program is performed in three phases: The first phase is improvement planning. It designs the improvement process, which is tailored to the goals and characteristics of the software organization. Depending on these goals and characteristics, the improvement process applies a specific combination of improvement methods, such as process assessments, measurement, quality management systems, knowledge management methods, technology transfer techniques, etc. The second phase of improvement management is the execution and control of the improvement process. In the third phase, the improvement program is analyzed and evaluated. This evaluation provides lessons learned and experience that can be useful for future improvement programs. It also shows to which extent the improvement program has reached its goals and triggers the setting of new goals for a subsequent improvement cycle.

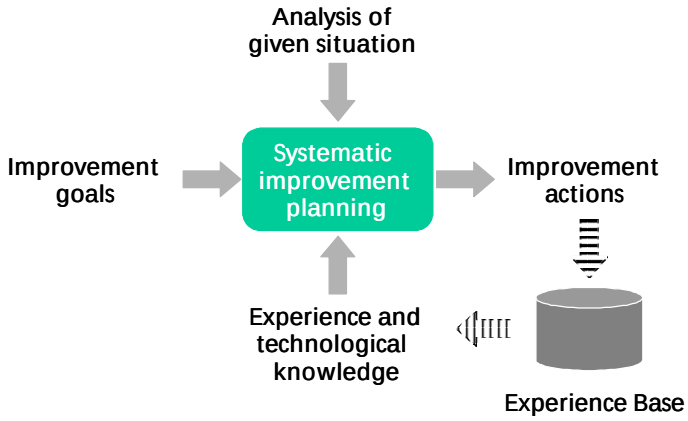


Figure 2: Experience-based improvement planning and experience accumulation

1.2 Systematic Improvement Planning and Tailored Improvement Processes

A particularly important step of systematic improvement programs is improvement planning. Figure 2 shows the basic principle of improvement planning: Improvement actions must be identified based on (1) explicitly defined improvement goals, (2) an analysis of the given situation (i.e., characteristics of the software project, the software organization, and their environment), as well as (3) experience and technological knowledge about what improvement actions are suited best to attain the improvement goals in the given situation.

Software engineering does not yet possess a particularly rich body of experience and knowledge about the effectiveness of improvement actions. This is mostly due to the short history of the discipline and its very dynamic nature. For this reason, particular efforts are required to obtain the needed experience. A suitable approach for this is the Experience Factory (EF) introduced by Basili et al. [2]. It guides software organizations through the process of accumulating and reusing important software engineering experience. Benefits of the Experience Factory are demonstrated by the success of the NASA Goddard Space Flight Center's Software Engineering Laboratory (SEL) [4].

Improvement processes must be tailored to the specific goals and characteristics of a software organization. Depending on these goals and characteristics, it focuses on different aspects of a software process or product, and it deploys different improvement methods to prepare and conduct the required process and product changes. Examples of improvement methods are: process assessments, measurement, quality management systems, knowledge management methods, and technology transfer techniques.

For implementing effective improvement processes, a software organization does usually need to establish appropriate improvement methods for three different key tasks (or *key practices*) of systematic improvement:

- Stabilization of work practices (e.g., through defined process models and software development standards)
- Intellectual control of software development
- Sharing and reuse of relevant knowledge and experience

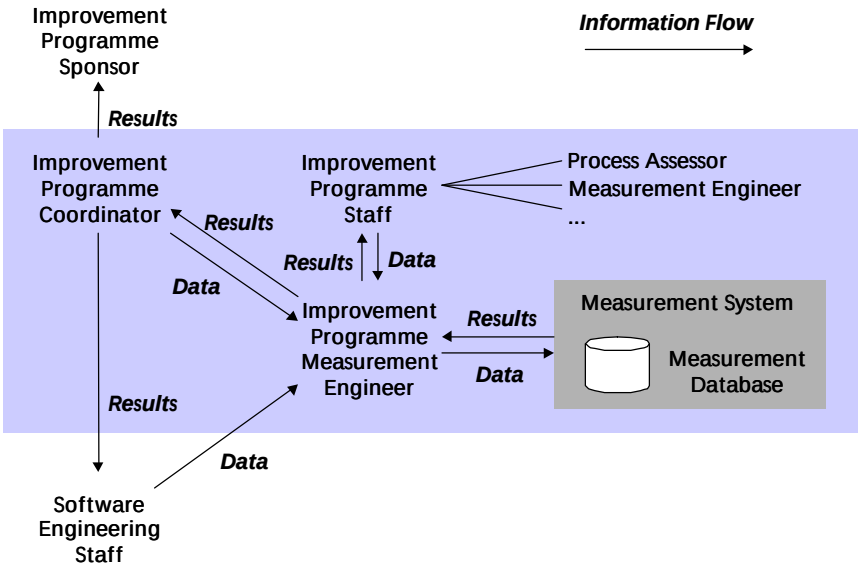


Figure 3: The organizational infrastructure for systematic improvement programs.

The most relevant improvement methods for the stabilization of work practices are modeling techniques such as process modeling and object-oriented modeling. Intellectual control of software development can be gained with the help of analytical techniques like process assessments, measurement, and simulation, which increase the understanding of a software organization's software development practices. Appropriate methods for sharing and reusing relevant knowledge and experience are the Experience Factory approach and the various techniques for knowledge management and organizational learning (cf. [1]).

1.3 An Organizational Infrastructure for Systematic Improvement Programs

The successful execution of an improvement program requires an appropriate organizational infrastructure, which ensures that all tasks are conducted in an effective and efficient manner. The typical organizational infrastructure of a systematic improvement program is shown in Figure 3. The improvement program team establishes a bridge between the sponsors of an improvement program (often higher-level management) and the software engineering staff of the projects in which the improvement program is performed. This team consist of a co-ordinator, technical staff for performing the relevant improvement techniques (e.g., process assessors, measurement engineers, trainers, QMS experts), and measurement engineer responsible for (meta-)measurement about the improvement program.

The organizational infrastructure should be complemented with an appropriate tool infrastructure. The core element of the tool infrastructure is the measurement database for data about the improvement process, which can be a database or spreadsheet application. This database is an important tool for monitoring and controlling the improvement program. In addition, tools are needed for data collection (e.g., paper forms or on-line questionnaires), for data analysis (again, database or spreadsheet applications), for data presentation, and for storage of measurement results (e.g., a document database or a web-based repository).

2 A Method for Defining and Controlling Customized Improvement Programs

Systematic improvement programs should be performed in three phases (cf. Figure 1):

1. Improvement planning
2. Improvement program execution and control
3. Improvement evaluation

The first phase, improvement planning, develops a tailored improvement process that is specific to the goals and context situation of the given software organization. It deploys selected improvement methods (e.g., process assessments, measurement, knowledge management etc.) so that the specific improvement goals can be attained in an efficient manner. The second phase, improvement program execution and control, applies this improvement process to one or more software projects. Finally, in the third phase (improvement evaluation), the success of the improvement program is evaluated and lessons learnt for future improvement programs are identified. This can trigger a new iteration cycle of the improvement program for which new goals are set or the improvement program is extended to other parts of the organization. An explicit management perspective on the improvement program is important to ensure its successful and efficient execution.

The following subsections explain each of the three phases of a systematic improvement program.

2.1 Improvement Planning

An improvement program is usually triggered by some kind of problem with the software organization's products or projects. This problem may have been perceived already or it may be anticipated for the future. In order to react to or prevent the problem, an improvement process needs to be planned in a three step procedure:

1. Analysis of the given situation
2. Identification and definition of improvement goals
3. Planning and adaptation of improvement process

The analysis of the given situation investigates the problem. The objective is to gain a first understanding of both its root causes and effects. The analysis addresses the organization's software development practices, relevant quality profiles of existing products, as well as important market characteristics and technological trends. It is recommended to conduct this analysis thoroughly but not at too much detail, because the improvement program has not really started yet. In particular, there might not be sufficient resources available for elaborated investigations. In the case that detailed analyses would be required, this need for information should be noted and addressed later as one of the first activities of the improvement process.

Based on the understanding of the initial situation, one or more improvement goals should be defined explicitly. They will from now on guide all subsequent steps of the improvement program. It is important that not too many improvement goals are addressed at one point in time in order to keep the improvement program focused. Improvement goals should refer to criteria that can clearly be identified in order to assess the improvement program's status or success. In

addition, it should be possible to attain the improvement goals within about one year. If am important improvement goals requires more time to be attained, it should be broken down into a set of goals that can be tackled in subsequent iteration cycles of an improvement program.

The actual improvement planning step identifies and prepares the measures needed for attaining the defined improvement goals. It defines an appropriate improvement process that integrates one or more improvement methods such as process assessments, process modeling, measurement, knowledge management, technology transfer, etc. These improvement methods can be categorized into analytical and constructive measures. Analytical measures provide the information needed for determining and implementing a change to the existing software engineering practices. Examples are process assessment and software measurement programs.

The determination and actual implementation of the change (or *improvement action*) is subject to the constructive measures of the improvement process. Examples of constructive improvement methods are process modeling, knowledge modeling and dissemination techniques, and the transfer of software engineering technology. Analytical measures can also be applied for monitoring and assessing the progress of the improvement program (e.g., a measurement program that monitors the intended effects of a newly introduced technology).

Additional tasks that must be accomplished during improvement planning are (1) gaining management commitment for the improvement program (i.e., from the sponsors of the improvement program), (2) the motivation of the project team members to support the improvement program, (3) the establishment of the improvement infrastructure in terms of organizational entities, personnel, and tool support, as well as (4) the detailed time and effort planning.

From the viewpoint of improvement management, another important task is the planning of checkpoints and measurement procedures for the improvement program. These checkpoints and the measurement provide the information needed for controlling the improvement program and for evaluating later whether it has been successful. An approach for the continuous measurement of improvement programs has been presented in [7].

2.2 Improvement Program Execution and Control

The execution of the planned improvement process should be accompanied by the managerial tasks of improvement program monitoring and control. So the second phase of the improvement program should include the following two tasks:

4. Execution of the improvement program

5. Monitoring and control of the improvement program

The improvement program is executed according to the previously planned process. In charge of the improvement program execution is the technical staff of the improvement team or external coaches or consultants. Depending on the selected improvement methods, these can be the process assessors, measurement engineers, technology transfer experts, etc.

The monitoring and control of the improvement program are under the responsibility of the improvement program manager. Monitoring is performed according to the planned measurement and assessment procedures. The measurement and assessment results are compared to the predefined target profile. If the actual data deviates from the target, the improvement program manager, together with the team, decide about possibly required corrective actions. If needed, the

improvement process must be redefined and, possibly, new commitment must be gained from the sponsor of the improvement program. The measurement and assessment results as well as all major decisions about the execution of the improvement program should be stored for the later evaluation.

2.3 Improvement Evaluation

At the end of the improvement program, or at major milestones of the improvement process, the improvement program should be evaluated. Experience gained through this evaluation should be packaged and disseminated so that the entire organization and future improvement programs can benefit from it. Improvement evaluation can be accomplished through the following two steps:

6. Analysis and evaluation of the improvement program

7. Packaging and dissemination of the gained experience

During the analysis and evaluation step, the tracked measurement and assessment data from monitoring the improvement program are reviewed and investigated. It is also recommended to perform a project post mortem review of the improvement program. Its participants should be the entire improvement team and the major collaboration partners from the related software projects (e.g., project managers, quality managers, senior engineers, etc.).

The analysis goals are: (1) Whether the improvement program was successful and the initially set improvement goals have been reached, (2) whether there is potential for further enhancement of the improvement program activities, and (3) which further improvement goals should be addressed in future improvement activities. If the improvement goals have not been reached, then the root causes should be analyzed and appropriate actions for actually attaining the goals should be initiated. Further enhancement of the improvement process can, for instance, concern the way in which the improvement team collaborated with the associated software projects. Other possible enhancements can refer to the effort and time estimations for improvement activities as well as the technical details or the integration of individual improvement methods (e.g., the integrated application of process assessments and related measurement programs).

In the final step of the improvement program, the analysis and evaluation results should, first, be put into action and, second, be packaged and made available to future improvement programs so that these improvement programs can benefit from the experience. Appropriate means for experience packaging are a company-specific improvement handbook, slide sets for the education of improvement teams, or a specialized repository of lessons learnt about improvement programs. In the long run, a software organization can benefit from the establishment of an experience base about experience from and for improvement programs [2] [4] [7]. It will be the source for the continued self-improvement of the organization's process improvement capabilities.

3 Scenarios of Tailored Improvement Processes

Many improvement programs are established to address a specific improvement need associated with product quality or a similar project performance measure (e.g., time to market, development cost, or productivity). An improvement program will be particularly successful and easy to manage, if these initial product-related improvement goals are made explicit and steer the activities of the improvement program (cf. the PROFES improvement method [13] [10]).

However, in larger companies improvement programs often need to be partitioned into smaller improvement projects that together help achieving the overall improvement goal. Usually it is hard to start with a large program that affects all development areas. The reason is that such a large program will consume too much resources, has a high risk of failing when obstacles appear, and project management will be difficult. As a consequence, after defining an overall improvement goal and a long-term strategy, it is best to identify pilot areas that will act as test beds for introducing improvement actions. Following the spirit of an experimental approach, such smaller projects offer large possibilities to learn about the pitfalls of improvement actions in the context of the given company. Using the knowledge gained in these experiments, the extension of the improvement program to new areas will be much easier, with reduced risk and less effort.

The next two chapters will provide examples of different improvement scenarios. In the first example, a company started with a small measurement program that was gradually extended to a larger improvement and knowledge management program. In the second example, a company decided to derive the improvement program from product-related improvement goals.

3.1 Corporate Improvement and Knowledge Management Program

In this chapter we present an example of a company with distributed software development that decided to start the improvement small, with a measurement program in a selected department at one location. The idea was to learn about the introduction of measurement, to collect the experience, and to expand the measurement program stepwise until a level was reached that would make it possible to run a full-scale improvement program. From the beginning it was clear that the introduction of knowledge management would be an important step to secure the results of the measurement program and support long-lasting improvement.

At start, a goal-oriented measurement program was set up following the Goal/Question/Metric approach that involved only a few dozen developers at one development site. Some previous attempts with measurement had failed and it took some significant effort to convince the developers to participate in the new program. The development of GQM-plans and the definition of the measurement process relied heavily on the participation of those people, and thus it was possible to create a positive spirit. One central aspect of GQM-based measurement is the interpretation of the measurement results in so called feedback sessions. Setting up such feedback sessions requires not only effort for the preparation of measurement results, but also consumes time, that is usually spent on project work. Convincing all involved persons that the time spent on feedback session is not wasted, but helps to better guide the project and finally achieve improvement took some time.

The initial goal of measurement was to understand the characteristics of the development process, to gain insights, and to create a baseline to track improvement with future measurement results.

After about one year the measurement program seemed to be settled enough in this department to start extending it to neighboring departments. Understanding as gained and it was possible to observe first positive effects in the pilot department. Improvement was initiated and satisfaction of all participants was rated high.

Moving to other departments required adapting the GQM-plans slightly. However, the experience gained over one year with the pilot department helped not only to come up with appropriate GQM-plans in a short time, but also reduced significantly the effort needed to establish the measurement process. Knowledge was already available on how to conduct feedback sessions

and how to prepare measurement results for the discussion. Evaluation tools could be re-used thus reducing further the effort for the extension of the measurement program.

The step-wise introduction of the measurement and improvement program required a set of role definitions that allowed staffing the project according to the current needs. At start only two members of a central department acted part-time as coordinators for the program, worked on the definition of the GQM-plans and the introduction of measurement in the departments. The Fraunhofer IESE provided additional methodological support. The introduction of feedback sessions made it necessary to have people on site who could conduct the sessions, help interpreting the measurement results, and act as interface between developers and project team.

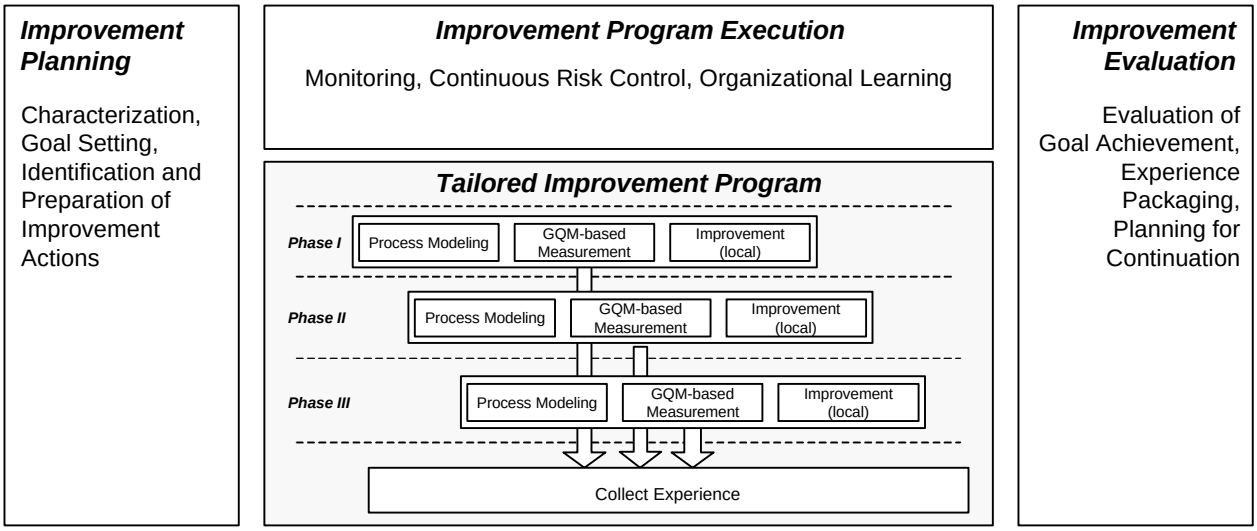


Figure 4: Tailored Improvement Program

After about 2.5 years the measurement program involves two product lines with several departments at different locations. A lot of experience has been gained, not only about the development process, but also about how to introduce goal-oriented measurement. As a result, the next step, started this year, is the implementation of a so-called "Experience Factory" that will collect the experience and make it available for future use.

At the time, processes for collecting, packaging, and storing experience are designed and will be implemented during fall 2000. The packages in the experience base will be either Lessons Learned, complementing the measurement results also available in the experience base, packages about theoretical issues or so-called best practices.

The experience base, which will act as central repository for the different types of experience, will be connected to the companies' Intranet, making access to packaged experience easy for all employees.

We have attended the improvement program over the last three years, helping the company setting up their improvement program, defining the measurement program, and extending the measurement program gradually to the current extend. The path that we chose was directly derived from our understanding of systematic improvement and the project helped us to better understand many aspects of systematic improvement programs. One task that has not been addressed so far in the project is the improvement evaluation. We are currently involved in defining an evaluation framework that shall be used at the company in the future.

Important experience we could derive from this project is about tailoring systematic improvement. It showed that having a concept for composing a company specific improvement program from a set of alternatives eased the discussion with responsible managers and allowed the company a more informed decision making.

3.2 Product Quality Driven Process Improvement

This section reports the example of a product quality driven improvement program. It has been performed at an embedded systems developer that is specialized in a specific technical domain. The improvement program was performed in a three years project that developed a new product generation. The new product contained much more software-based functionality than the previous ones.

The project was performed in three incremental phases. Already the first increment contained the core functionality and was subject to field tests. The peek staff count of the project was about 60 persons, most of them located at the same development site. Part of the system functionality was acquired from or developed by subcontractors.

In the first step of the improvement program, the improvement goals were defined. These were (1) to attain high product reliability (i.e., low number of faults detected in the field) and (2) to assure in-time delivery of the product increments. These goals were viewed important for the successful launch of the new product generation. They also were interrelated with each other, because if there were too many defects late in the project, the additional defect removal effort would put the product delivery date at risk. Likewise, a too high time pressure throughout the project might cause higher defect rates than normal.

All subsequent steps of the improvement program were to ensure that these two improvement goals were attained. Therefore, the development process had to be designed appropriately and the project status had to be monitored continuously with regard to these goals.

The second step of the improvement program was to establish software engineering practices that helped ensuring the attainment of the improvement goals. The initially established process changes were a strong emphasis on the system and software requirements analyses of the first increment (i.e., dedicating much time and resources to the specification of the product), the introduction of a formal inspections for requirements, architecture, and design inspections, the definition of new test processes, as well as the installation of a new configuration system. These initial process changes were followed later by additional change actions that resulted from the measures of the improvement program.

After the initial process had been established, a process assessment was performed. It provided baseline for the later evaluation of process improvements. It also recommended further process changes. Most of them were implemented later in the improvement program. Examples are additional defect management practices, subcontractor management processes, and a new integration process.

As the third step a measurement program was set up following the Goal/Question/Metric (GQM) method [3] [8][9] [15]. The GQM goals were defined so that the most central improvement goal—i.e., the attainment of product reliability—could be monitored through the measurements. In addition, the new inspection process was measured. It was supposed to be the most important improvement action for ensuring low defect rates.

The measurement program actually increased the understanding of the inspections' effectiveness. This created deeper trust into the new process at the management level and among the engineers. It also assisted the further adjustment of the inspection processes to the project's needs. The measurement of the workproducts' defect rates (i.e., related to the improvement goal *reliability*) was used primarily by project management to estimate and control the product delivery date. It later developed into the core of a new defect management process.

Already for the first product increment the improvement goals were attained. This could be repeated also for the two other increments. Gradually, the capability maturity of the development processes as well as the quality of the developed product were increasing. There was convincing evidence that these effects were largely resulting from or facilitated by the various activities of the improvement program.

After the first product increment had been delivered, an overall analysis of the measurements was performed by the management and senior team members. In addition, a second process assessment was conducted. These activities resulted in adjustments of the measurement program (i.e., some measures were re-defined and additional ones included) and in the identification of additional process changes for the next phase of the improvement program.

As the main actor of the improvement program, a senior engineer was working half-time managing the improvement program. He was supported most of the time by one or two junior engineers who were working between 50 to 100 per cent of their time for the improvement programme. In addition, external consultants performed the process assessments and coached the measurement program. Management support was very high, because the improvement program was viewed a major means to ensure project success.

The GQM measurements and the process assessments did very well complement each other. The process assessments provided the project team with a broad overview of the project's software development capabilities. The measurement program deepened the understanding of the most relevant product quality aspects and monitored the progress of selected process changes during the time between two process assessments.

Particularly beneficial for the progress and the manageability of the improvement program were the strong focus on explicitly defined product quality goals and the continued assessment of goal achievement throughout the improvement program. It provided a clear rationale for the improvement program both to management and the project team. The goals also made it easy to select effective improvement actions. This reduced the overhead effort needed for improvement and led to fast improvement results.

4 The Role of Knowledge Management in Systematic Improvement

We consider improvement as a systematic and continuous endeavor to reach a higher customer satisfaction through delivery of products that meet given requirements. Understanding what quality is and how it can be improved is a learning process that never ends [15]. Consequently the experience made in this process should be collected, stored, and used for all upcoming activities. The better a company can capitalize on the lessons it has learned from its operation, the higher are potential benefits on the market (higher customer satisfaction, shorter reaction times, higher savings, or better quality). In the light of this definition it should be clear that for us knowledge management is a vital part of systematic improvement.

When talking about the relationship between knowledge management and systematic improvement the following aspects should be clearly separated.

(1) The introduction of knowledge management can directly profit from systematic improvement projects, because essential knowledge is revealed during the course of such projects. Process modeling is a good example of an activity that is usually carried out in the early phases of an improvement program, but hardly for knowledge management purposes only. The resulting process models make knowledge about activities (practices) explicit.

(2) Closely related to the facilitation of knowledge management programs is the fact that knowledge is needed **for** improvement. Understanding processes and products is the first step in improvement and knowing how to act best in a given situation to achieve a goal makes the difference between successful and unsuccessful improvement programs. This type of knowledge helps a company to improve its software projects directly. It is knowledge not only about problems and strategies developed to resolve them, but also experience about the effectiveness of actions acquired when applying the strategies. Usually (without knowledge management) this knowledge and experience is hardly recorded and preserved for future use, thus getting lost after a short time.

(3) Finally, improvement itself consists of many projects. Consequently, there is also learning about conducting better improvement projects. We call the knowledge gained in this context knowledge **about** improvement. It helps to develop skills in improvement management and allows apply learning also to this level of operation. This type of knowledge helps a company to improve its software projects only indirectly, but is in our opinion as essential as the type of knowledge mentioned above.

In any case the setting up of knowledge management means to create the infrastructure and the spirit that helps collecting and sharing knowledge within and across (any type of) projects. This task is part of the planning and execution of the improvement program and requires the same thoughtful tailoring as the improvement process itself.

We would like to come back to the example we presented Section 3.1. In this case the introduction of feedback sessions and the implementation of the Experience Factory are part of a company-specific knowledge management initiative. While the feedback sessions help to improve the direct sharing of (tacit) knowledge, the Experience Base will contain explicit knowledge about the development process. The Experience Factory processes that will be introduced aim at eliciting and packaging knowledge and distributing it to all potential users.

In Section 3 we have not presented an example strategy or scenario for learning about improvement. But, as stated above, using knowledge about improvement will help a company to »improve improvement«, i.e., initiate and conduct improvement projects more efficient and avoid mistakes. The most severe problem in this case is, that it takes a significant time before a critical mass of such knowledge is build up that really provides support for the different steps of an improvement program. Consequently the Fraunhofer IESE together with the Fraunhofer Center Maryland has initiated a project that brings together companies from the IT sector to share knowledge about improvement across company boundaries.

The so-called Software Experience Center (SEC™) is a consortium that started work in 1999 and today brings together five companies from Europe and the US for the purpose of an open experience exchange and for setting up and performing joint case studies. The main goal is to promote the extension of Learning Organizations concepts to the software domain. The

international set-up is expected to create insight into Learning Organization issues across different cultural environments.

Collection and dissemination of experience will be performed mainly by means of workshops, reports, and an Experience Base, which will provide enough detailed information for the SEC member companies to directly utilize this experience in their own organizations to:

- significantly speed-up each company' s internal improvement activities
- increase the speed of learning across the company
- avoid costly technology transfer problems and other types of crucial mistakes
- include technology advances in strategic planning

The Fraunhofer Institutes act as facilitators and bring added value to the SEC consortium. In particular they:

- plan, coordinate, and execute SEC workshops
- contribute tutorials, technology presentations, and experience reports to the workshops
- collect experience in the course of bilateral projects with members and document it for dissemination within the consortium
- maintain the SEC consortium's Experience Base, which makes the consortium's experience assets accessible to the members

Experience from about 1.5 years of operation shows that there is a large interest in learning about improvement, but that at the same time many obstacles are to overcome when experience shall be shared among companies.

5 Summary

This paper has introduced a method for the systematic management of improvement programs. It places particular emphasis on the definition and control of improvement programs that are customized to the specific goals and characteristics of a software organization. The method is illustrated and supported by scenarios of systematic improvement programs that have been conducted at two industrial European software organizations. Knowledge management plays a particularly important role for the long-term benefit and success of improvement initiatives.

An important experience from applying the described systematic improvement method is that an improvement program should be started small with a few improvement goals and limited to a part of the organization. This has several advantages. It is often much easier to get funding for smaller programs that have no cross-departmental impact. Less coordination is needed and the chance of finding sponsors for the improvement actions is much higher. The experience that we reported in Section 3.1 underlined this thesis. Using a limited measurement program was beneficial when convincing department heads to provide time and money. Several problems could be solved with limited effort in small teams, using the results later when extending the measurement program. The idea to collect, store, and reuse the knowledge that has been collected in the course of an (stepwise extended) improvement program came as a natural effect, when all participants recognized the value of the experience made in the last three years. At this point in time it was much easier to convince people to invest in further activities and to capitalize on they knowledge.

However, it is essential to have a clear idea about the general context, provide visions of a comprehensive improvement program and long-term goals. When tailoring improvement actions for a specific company, our framework for systematic improvement programs provided us with the guidance needed.

The importance of the systematic, goal-driven planning of improvement programs is supported by the experiences from the PROFES project[1] [13] [10] [6]. This European applied research and technology transfer project has developed a product quality focused process improvement method, which is a predecessor of the presented systematic improvement method. Experience from three industrial software organizations in the embedded systems domain have shown that goal-driven improvement programs can be particularly successful. For instance, relevant improvement effects can be attained very fast and with little overhead effort for the related software projects.

Future work at Fraunhofer IESE in the area of systematic improvement will focus on the further support of improvement processes for specific types of software organizations (e.g., SMEs or embedded software development) as well as on the accumulation and dissemination of experience about systematic improvement [5]. The PROFES project has developed a repository that contains effort models of improvement methods (e.g., process assessments and GQM measurement) [12]. Another PROFES repository collects information about the effects and application prerequisites of software engineering technology [11]. This information supports the goal-driven selection of improvement actions during improvement planning. Both repositories can be accessed through the internet. Also the Software Experience Center (SEC) described in Section 4 is a major line of future activity. Results from the SEC will gradually be shared and discussed with the public in order to develop a growing body of consolidated software engineering experience.

6 References

[1] K.-D. Althoff and W. Müller. Proceedings of the 2nd Workshop on Learning Software Organizations. Fraunhofer IESE, Kaiserslautern, Germany. (http://www.iese.fhg.de/LSOworkshop2000)

[2] V.R. Basili, G. Caldiera, and H.D. Rombach. Experience Factory. In J.J. Marciniak, ed., Encycl. of SE, vol. 1, pp. 469–476. John Wiley & Sons, 1994.

[3] V.R. Basili, G. Caldiera, and H.D. Rombach. Goal Question Metric Paradigm. In J.J. Marciniak, ed., Encycl. of SE, vol. 1, pp. 528–532. John Wiley & Sons, 1994.

[4] V. Basili, M. Zelkowitz, F. McGarry, J. Page, S. Waligora, and R. Pajerski. SEL' ssw process-improvement program. IEEE SW, 12(6):83–87, Nov. 1995.

[5] A. Birk. A knowledge management infrastructure for systematic improvement in software engineering. Doctoral dissertation, University of Kaiserslautern, Kaiserslautern, Germany, 2000. (to appear)

[1] ESPRIT Project No. 23236, PROFES, has been supported by the CEC. The project results can be accessed at the web site www.profes.org.

[6] A. Birk, P. Derks, M. Elf-Mattila, J. Hirvensalo, R. van Solingen. Product-focused software process improvement: The PROFES methodology and experience from its industrial application . Proceedings of the SQM'99, Cologne, 1999.

[7] A. Birk, D. Hamann, and S. Hartkopf. A framework for the continuous monitoring and evaluation of improvement programmes. In: F. Bomarius, M. Oivo, Proceedings of the Second International Conference on Product-Focused Software Process Improvement (PROFES2000), Lecture Notes in Computer Science, Springer, Berlin, 2000.

[8] L.C. Briand, Ch. Differding, H.D. Rombach. Practical Guidelines for Measurement-Based Process Improvement. Software Process Improvement and Practice 2 (4), pp. 253-280, 1996.

[9] F. van Latum, R. van Solingen, M. Oivo, B. Hoisl, D. Rombach, and G. Ruhe. Adopting GQM-based measurement in an industrial environment. IEEE Software, 15(1):78–86, 1998.

[10] The PROFES Project. http://www.profes.org.

[11] The PROFES Consortium. The PROFES Repository of Product/Process Dependency (PPD). http://www.iese.fhg.de/profes/PPDRepository.

[12] The PROFES Consortium. The PROFES Cost/Benefit Repository. http://www.iese.fhg.de/profes/CBRepository

[13] The PROFES Consortium. PROFES User Manual. Fraunhofer IRB Verlag, Stuttgart, Germany 2000.

[14] R. van Solingen, E. Berghout, The G/Q/M Method. McGraw-Hill. London, 1999.

[15] R. van Solingen, E. Berghout, R. Kusters, J. Trienekens. No Improvement without Learning: Prerequisites for Learning the Relations between Process and Product Quality in Practice. Proceedings of the PROFES 2000, Oulu (FIN), Springer LNCS 1840, pp. 36-47, 2000.

QWE2000 Vendor Technical Presentation VT4

Hans Buwalda
(CMG)

TestFrame: Getting Testing and Test Automation Under Control

Key Points

- Point 1...
- Point 2...
- Point 3...

Presentation Abstract

Abstract

About the Speaker

Hans Buwalda is project director at CMG, a leading European information technology services group. He is responsible for new developments around the TestFrame approach for testing and test automation of which he is the main architect. The approach has been started by him in 1994. In 1996 he presented the main ideas for the first time to an international audience in a speech called “Testing with Action Words, abandoning record and playback”. Since then the method is being used in an increasing number of countries and Hans has become a frequent speaker at industry conferences, tutorials, and workshops.

TESTFRAME

Getting testing under control

Hans Buwalda

TestFrame Research Centre

hans.buwalda@cmg.nl

TESTFRAME

Agenda

- managing testing and test automation
- our approach to testing
- implementation
- application areas
- TestFrame and CMG

CMG ++ TESTFRAME

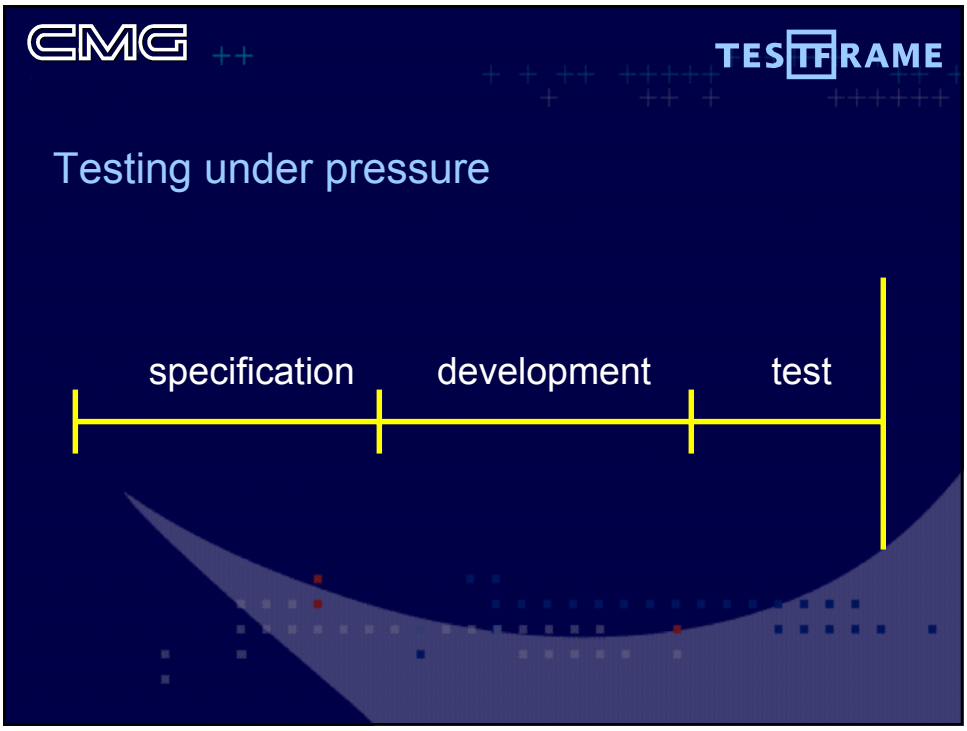
Why should we test?

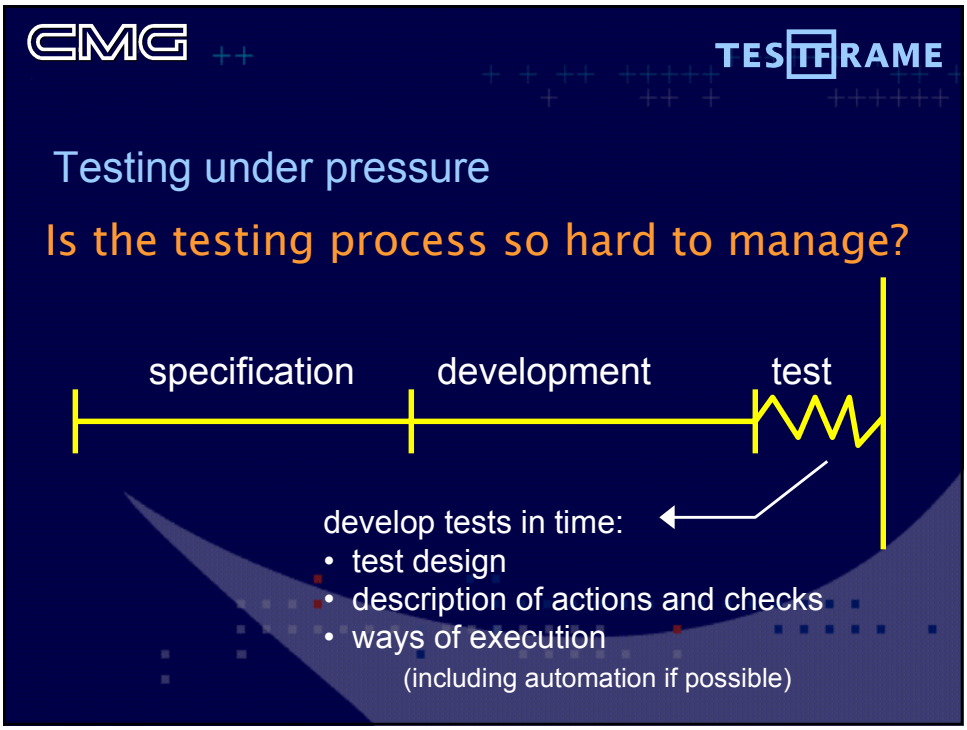
- people expect us to do
- somebody wants want us to
- increasing certainty and control
  - showing absense of problems
- finding faults saves money
  - showing presence of problems

CMG ++ TESTFRAME

Why should we not test?

- it costs time and money
- you might find problems . . .
- we forgot to plan for it
- we need the resources for development
- it is difficult
- and . . . it is hard to manage


CMG
TESTFRAME
Testing under pressure
specification
development
test


CMG
TESTFRAME
Testing under pressure
Is the testing process so hard to manage?
specification
development
test
develop tests in time:
• test design
• description of actions and checks
• ways of execution
(including automation if possible)

CMG ++ TESTFRAME

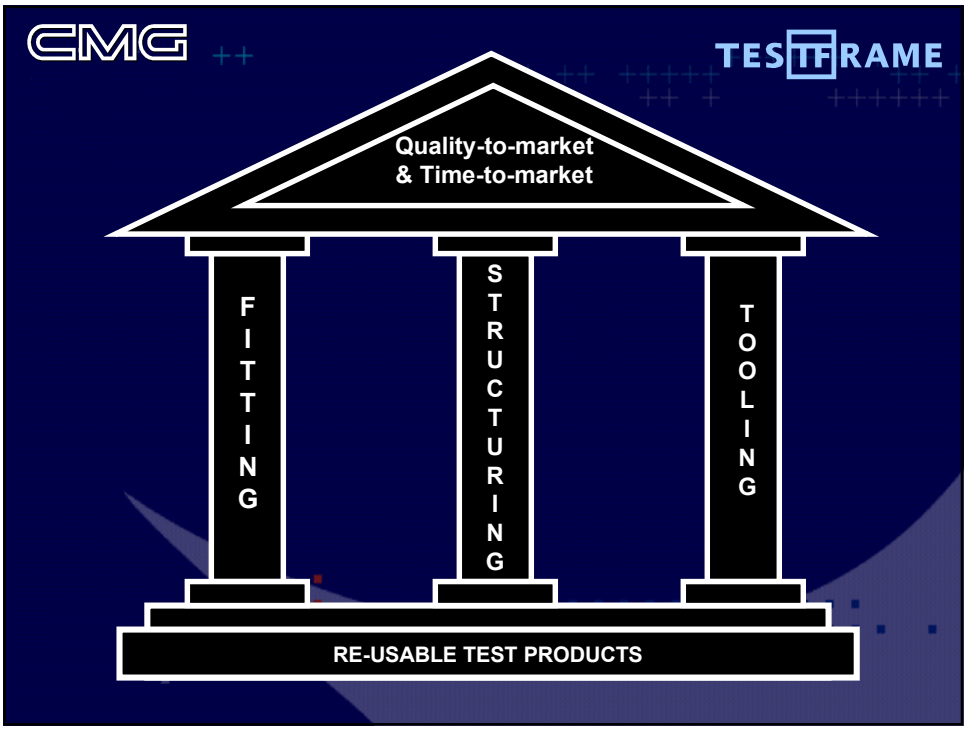
Why automate tests?

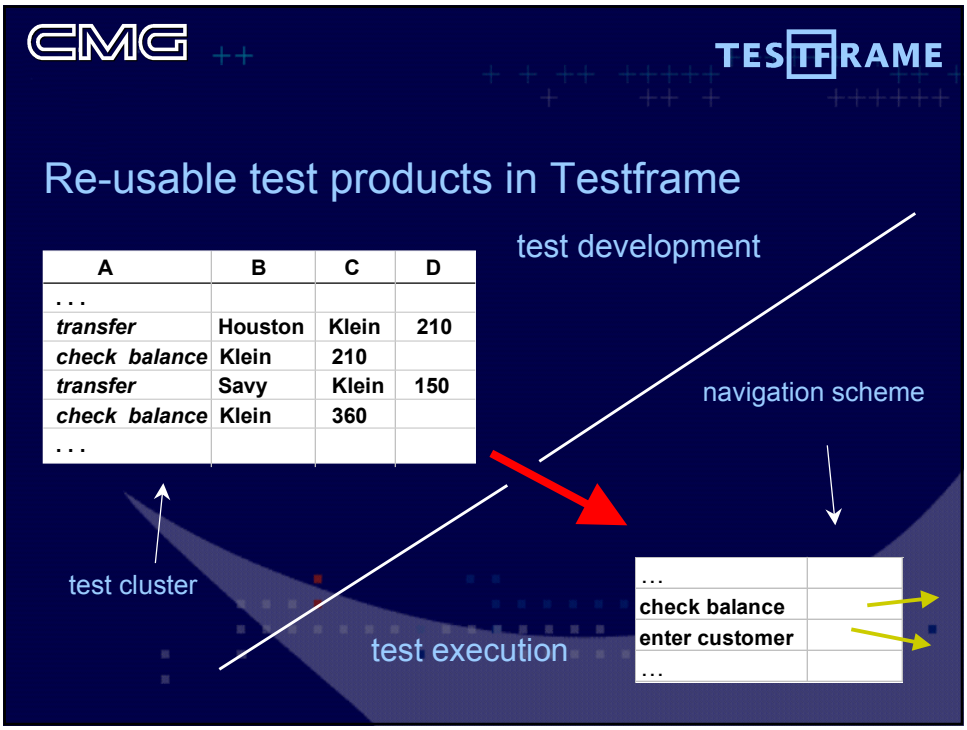
- it is more fun
- we don't have to do it ourselves then
- saves time and money
- reduces involvement of valuable specialists
- it can consolidate a structured way of working

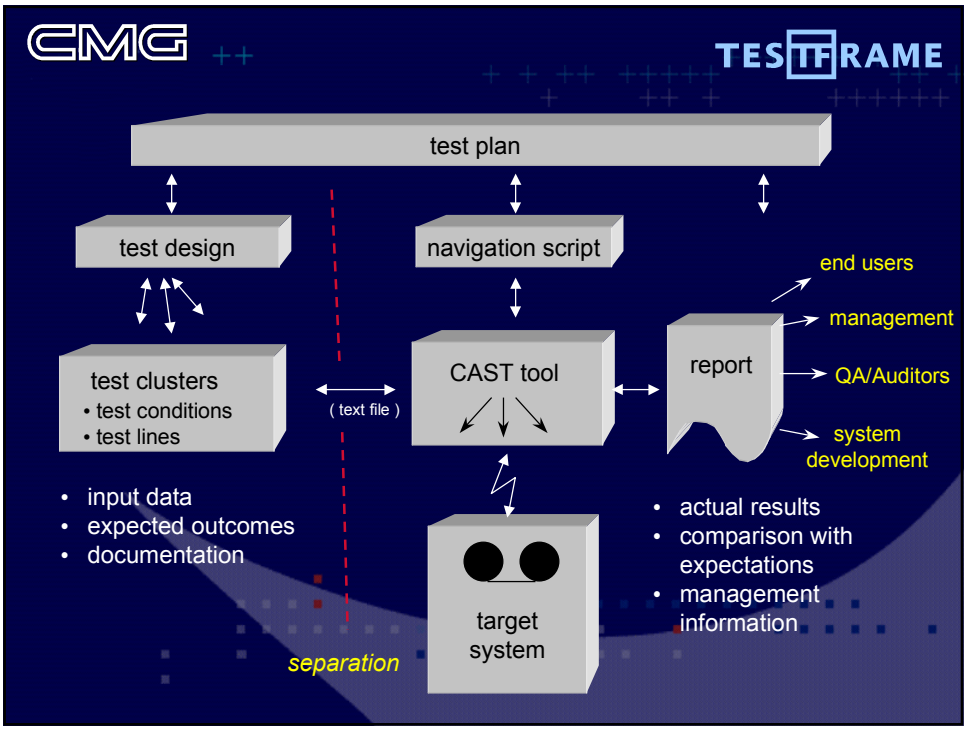
CMG ++ TESTFRAME

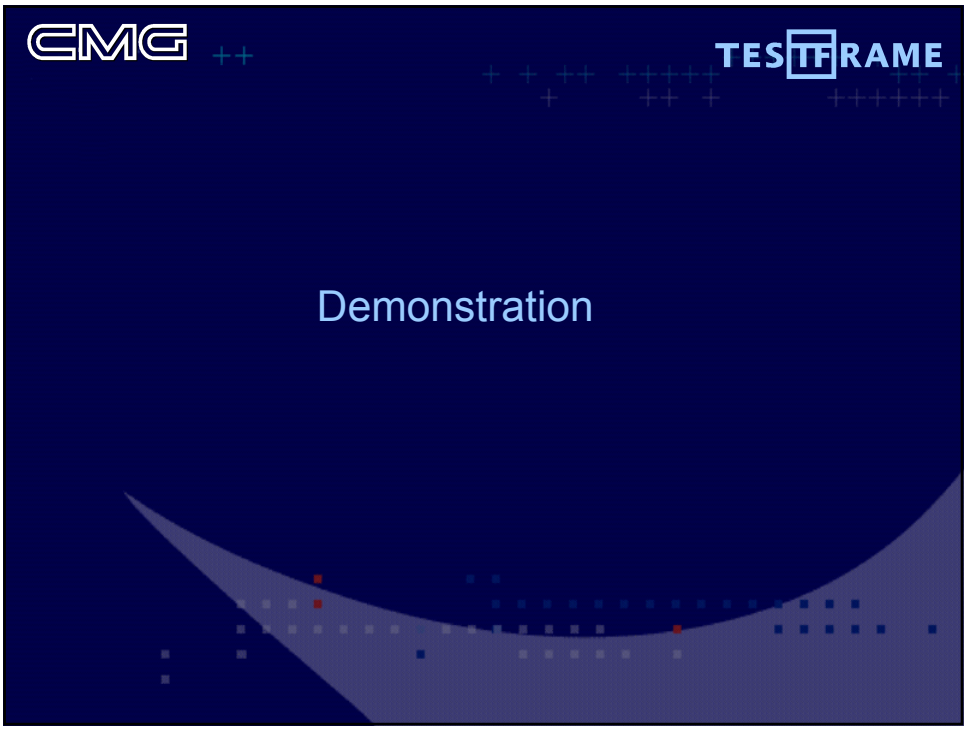
TestFrame

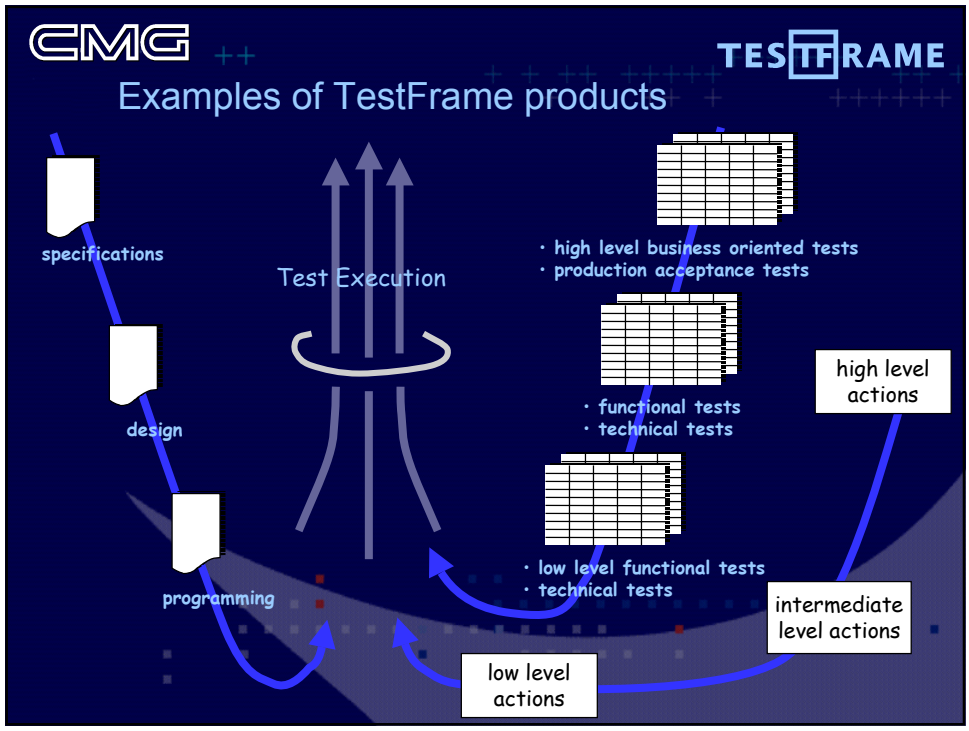
- established in 1994
- large scale use
- in industry, government, finance, telecommunication, publishing, ..
- continuous extensions

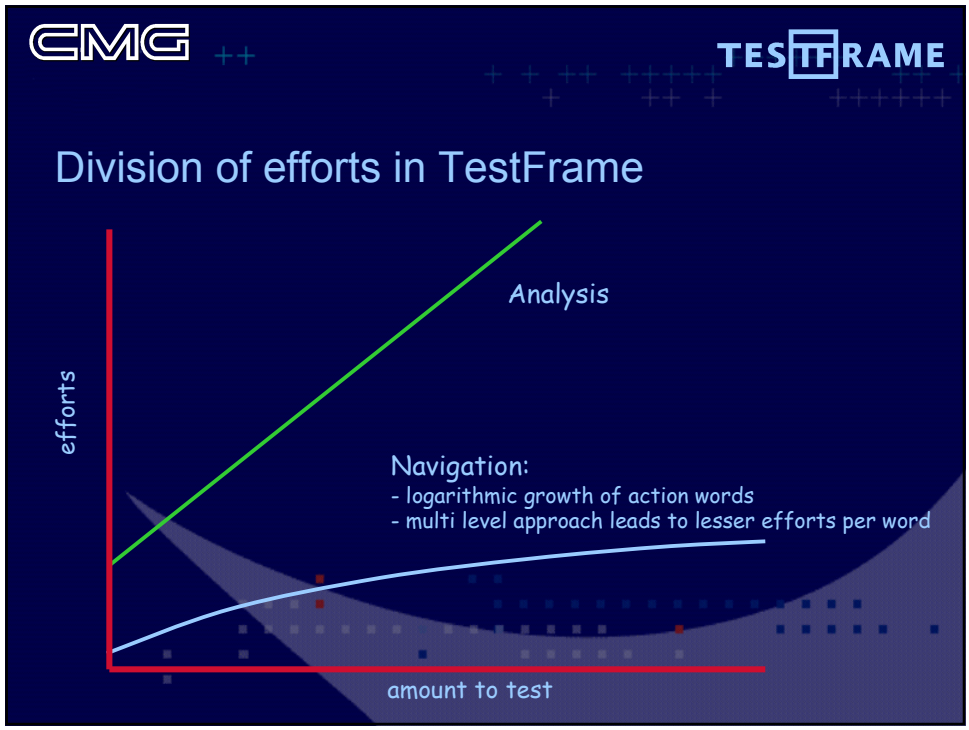

CMG
TESTFRAME
Quality-to-market
& Time-to-market
FITTING
STRUCTURING
TOOLING
RE-USABLE TEST PRODUCTS

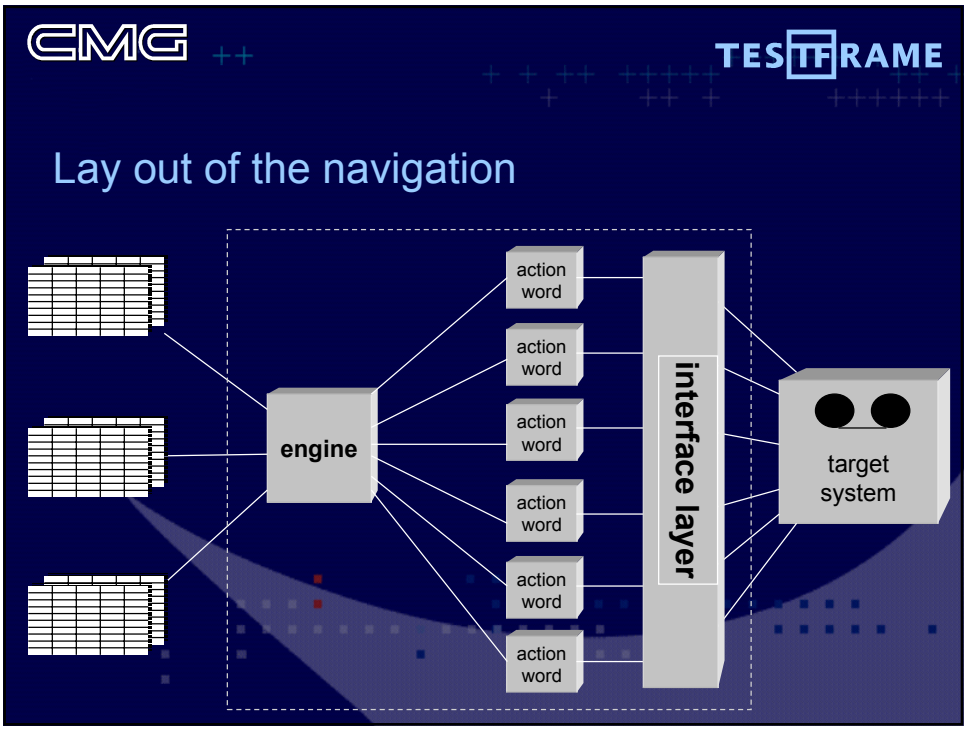

CMG
TESTFRAME
Re-usable test products in Testframe
test development
A | B | C | D
. . .
transfer | Houston | Klein | 210
check balance | Klein | 210
transfer | Savy | Klein | 150
check balance | Klein | 360
. . .
navigation scheme
test cluster
…
check balance
enter customer
…
test execution

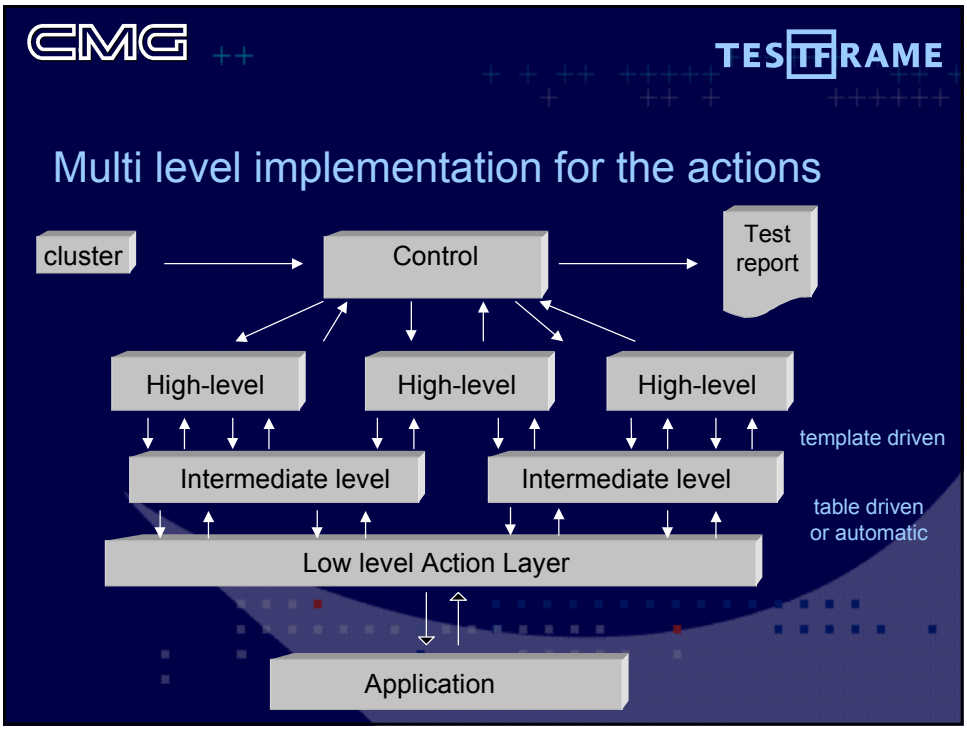

CMG
TESTFRAME
test plan
test design
navigation script
end users
management
test clusters
• test conditions
• test lines
(text file)
CAST tool
report
QA/Auditors
system development
• input data
• expected outcomes
• documentation
• actual results
• comparison with expectations
• management information
target system
separation

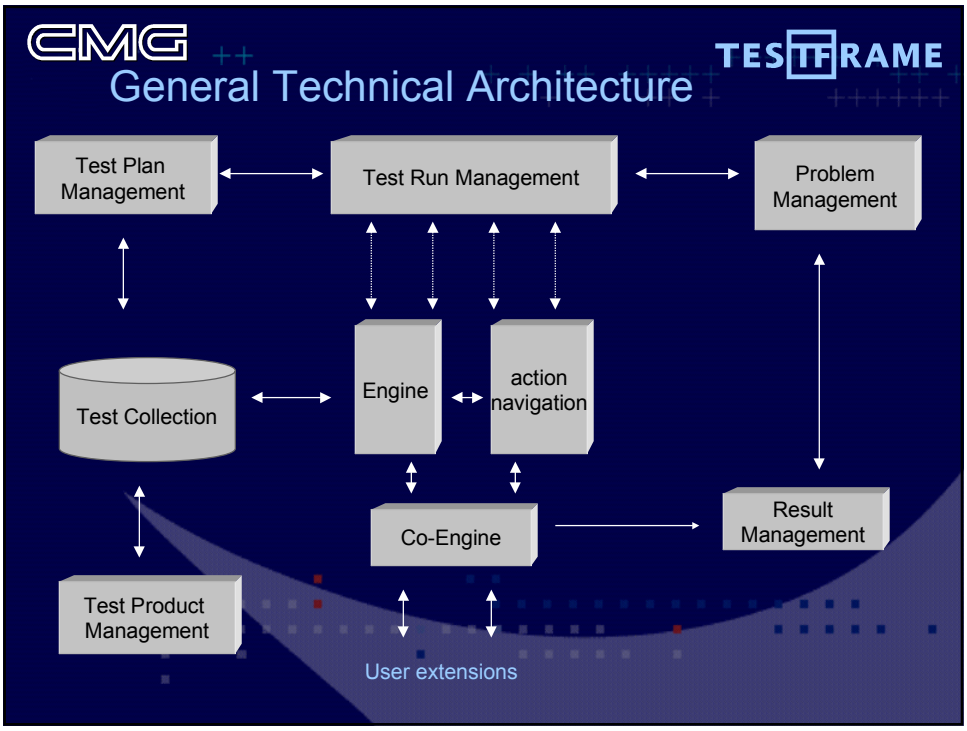

CMG
TESTFRAME
Demonstration

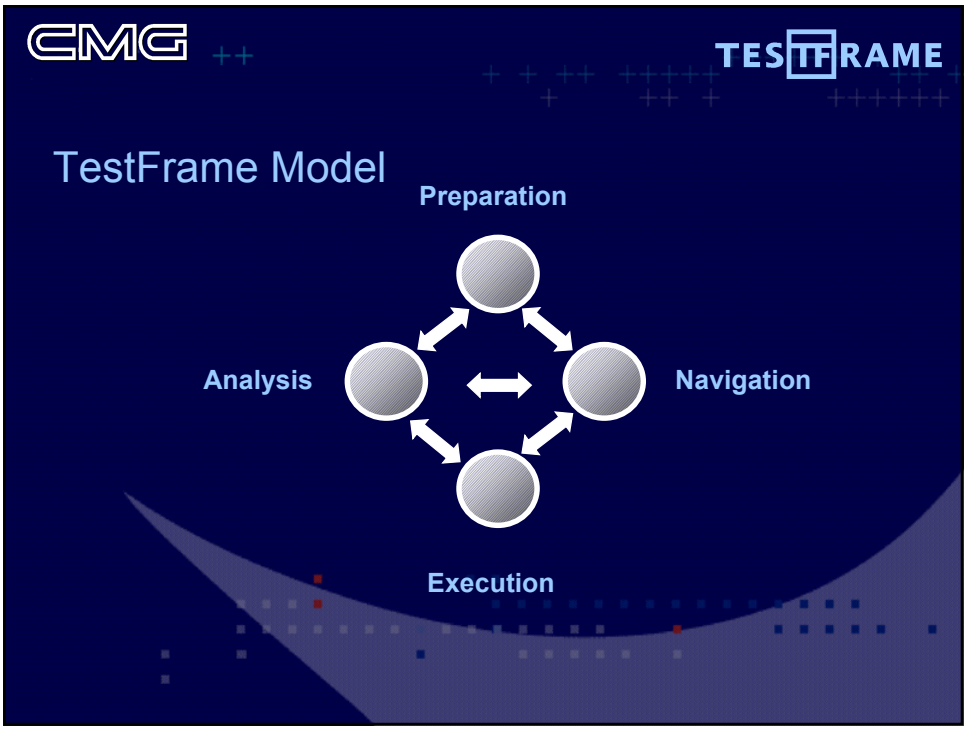

CMG
TESTFRAME
Examples of TestFrame products
specifications
design
programming
Test Execution
· high level business oriented tests
· production acceptance tests
· functional tests
· technical tests
· low level functional tests
· technical tests
high level actions
intermediate level actions
low level actions


CMG
TESTFRAME
Division of efforts in TestFrame
efforts
Analysis
Navigation:
- logarithmic growth of action words
- multi level approach leads to lesser efforts per word
amount to test


CMG
TESTFRAME
Lay out of the navigation
engine
action word
action word
action word
action word
action word
action word
interface layer
target system

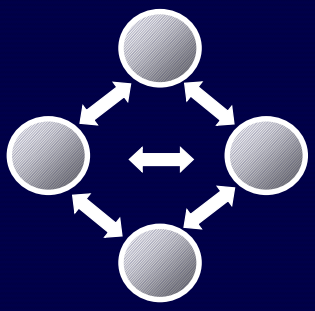

CMG
TESTFRAME
Multi level implementation for the actions
cluster
Control
Test report
High-level
High-level
High-level
template driven
Intermediate level
Intermediate level
table driven or automatic
Low level Action Layer
Application


CMG
TESTFRAME
General Technical Architecture
Test Plan Management
Test Run Management
Problem Management
Test Collection
Engine
action navigation
Co-Engine
Result Management
Test Product Management
User extensions


CMG
TESTFRAME
TestFrame Model
Preparation
Analysis
Navigation
Execution

Products

Preparation
- Plan of Approach
- Test Strategy
- Risc Analysis



Analysis
- Clusters
- Test conditions
- Test cases

Navigation
- Technical design
- Implementation Action Words

Execution
- Findings
- Test Report
- Evaluation

Management and control

- testing is a strategical issue
- testing improves control
- regard test automation as an “amplifier”
- several issues need attention to improve manageability of the testing process itself

CMG ++ TESTFRAME

Ontwikkeling van clusters

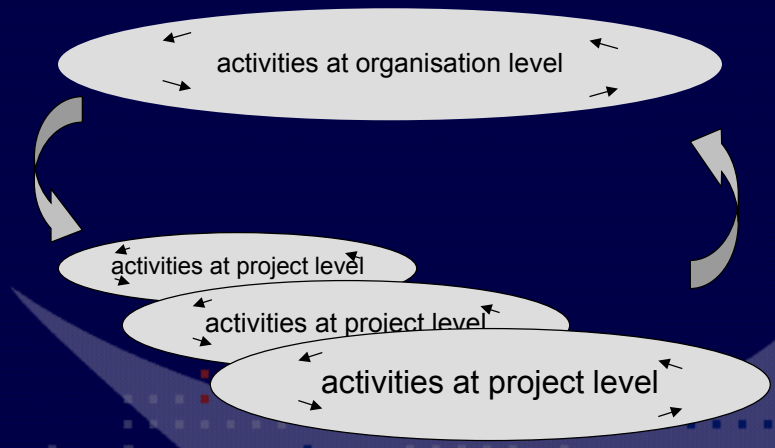
- risico factors:
  - business risks
  - technical risks
  - project risks
- input from stake holders
- relate to the planning of system development
- pay attention to issues in test execution

CMG ++ TESTFRAME

The testing regime

- embedded in development and maintenance processes
- re-usable well designed test products
- stable test automation
- incremental development
- well managed test environments

Organisation and project level



Activities at organization level

- One or more pilots
- Training and handbooks
- Resourcing (pooling, hiring)
- Auditing and reviewing
- R&D
- Development of common products
- ...

CMG ++ TESTFRAME

Proven benefits

- accessible and maintainable test structure
- early start saves time at the end of the project
- involvement of the expert users needed only once
- better control over the test process

CMG ++ TESTFRAME

Demonstration 2 (if time permits)

"Multicom"

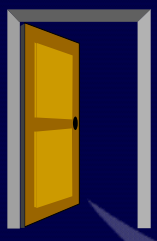
CMG ++ TESTFRAME

Application areas

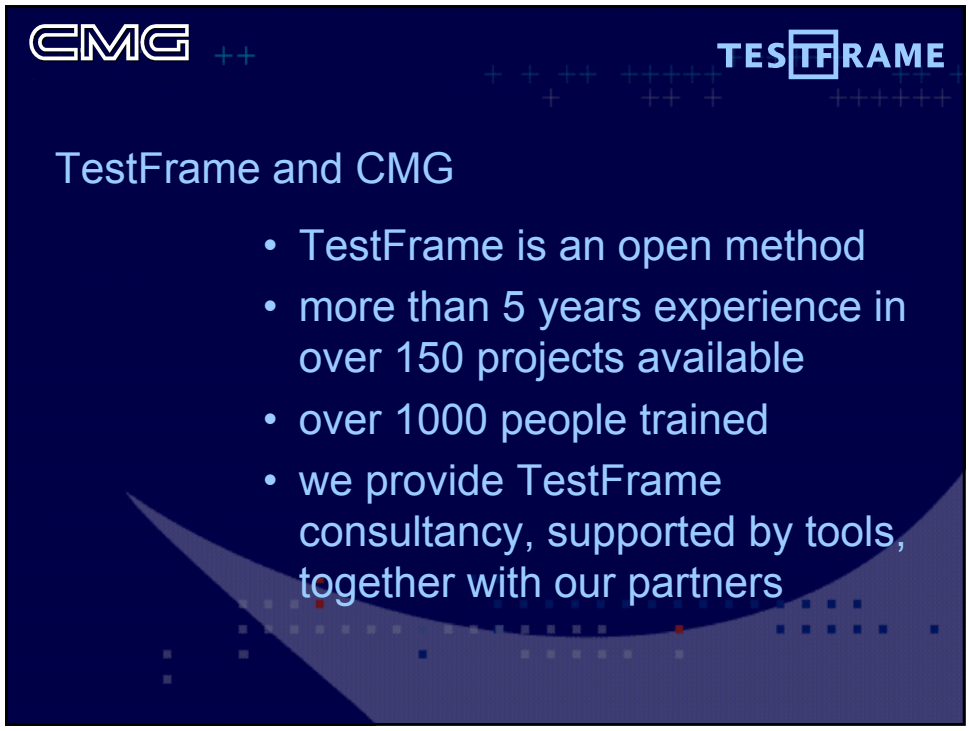
- Industry, Finance, Telecommunications, Public Sector, ...
- web, mainframe, C/S, ERP, technical and embedded software, infrastructure, ...
- not dependent on specific test tools

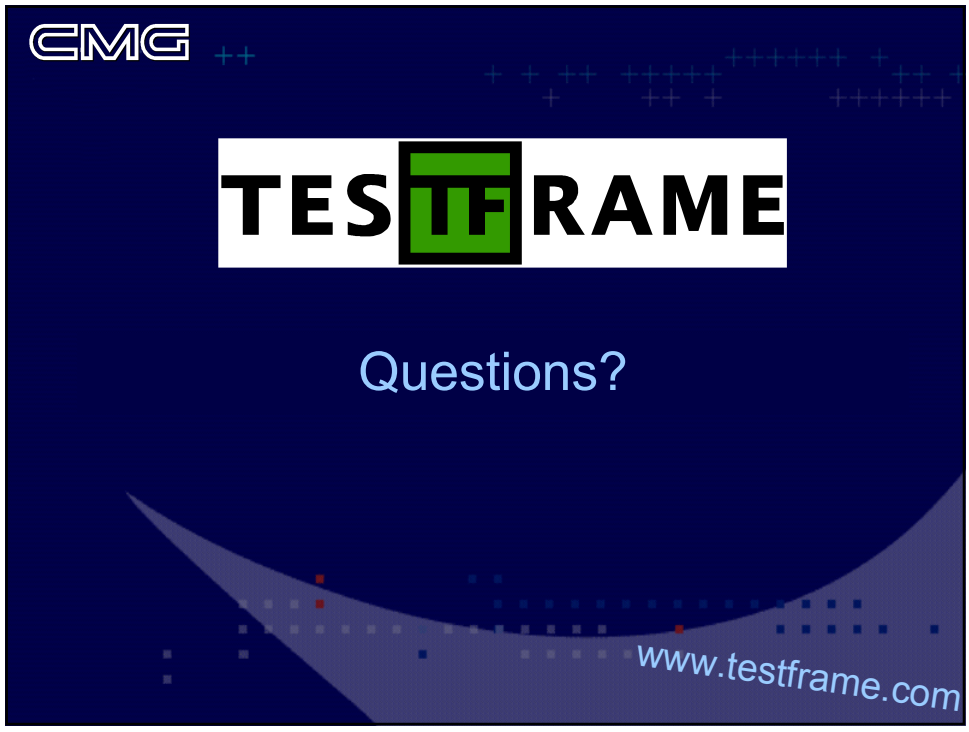
CMG ++ TESTFRAME

Introducing TestFrame



- see it as an organizational change
- use pilots, training and coaching
- no standard recipe, depends on:
  - skills available
  - experience with testing and test automation
  - time (and budget) constraints
- coaching and support is essential (there are many lessons to learn)


CMG
TESTFRAME
TestFrame and CMG
• TestFrame is an open method
• more than 5 years experience in over 150 projects available
• over 1000 people trained
• we provide TestFrame consultancy, supported by tools, together with our partners


CMG
TESTFRAME
Questions?
www.testframe.com

QWE2000 Session 5T

Ruud Teunissen
(Gitek)

Improving Developer's Tests

Key Points

- Why improving the developer's tests
- How to improve the developer's tests a practical approach

Presentation Abstract

This paper discusses the reasons for improving the developers' tests. Next, a practical approach is given for improving the quality of the developers' tests (i.e. program test and integration test). The emphasis of the approach is on:

- clear testing responsibilities;
- the idea that improvements come from within the development team;
- insight into the quality of the test object by using exit- and entry-criteria.

The paper concludes with two case-stories.

Introduction

Testing theory states the importance of early testing. The developers perform one of the earliest forms of testing. There is sufficient literature on how this kind of testing should be performed, including a lot of techniques and tools. However, there is a large gap between theory and practice. In practice, developers often test based on their intuition and in an unstructured way, in stead of using techniques, etc. Based on our own experiences, this paper gives reasons why better developer testing is important and how improvements can be implemented. The paper concludes with two case-stories. Our experiences originate from the world of administrative automation (financial institutions, government, industry), and "new media" projects (such as internet, knowledge management, data ware housing).

Characteristics of developers' tests

Typical developers' tests are the program test and the integration test. Some characteristics of these tests (and differences with other types of tests) are:

- The finder of a defect is also (often) the solver of the defect. This means communication overhead can be kept to a minimum;
- (In principle) all defects found should be repaired before the product is

handed over. Therefore, the amount of reporting is limited;

- A developer differs in attitude from a professional tester; the former wants to demonstrate that the product works, the latter wants to demonstrate the inverse. Time-consuming and thorough testing conflicts with the developer's attitude;
- The tests are an integral part of the development process;
- At the time these tests start, most defects are in the product: this requires cheap and fast repair of defects.

Should developers' tests be improved?

When asking the developers whether their ways of testing need improvement, frequent answers are:

- Not enough time, too expensive;
  Only if given more time and money, better testing is possible. However, development is always short of time and money. But when the project leader is asked for more time and money, that person responds something like: "If I give the developers more time and money, they'll certainly spend it. The big question is, on what?")
- Faith in current way of working, in current product quality;
  It is common knowledge that 100% defect free software is impossible to achieve. Apart from that, developers are proud of the products they develop.
- A good (better) test follows.
  Test professionals perform later, better tests. These people are trained in testing and they typically find a lot of defects. And what's more important, they like testing. Because for most developers à
- Testing is boring!

There are of course a number of reasons for improving developer testing:

- Defects one finds oneself are easier to analyse than defects found by other parties;
- Early rework is cheaper, since knowledge is still fresh and all relevant parties are still there;
- Earlier feedback prevents similar errors;
- White-box techniques used in developers' tests find different defects from black-box techniques used in later tests.
- Because higher quality products are delivered, fewer defects are found in later tests (and in production). As a consequence, developers can spend less time reworking products and more time creating products;
- One of the most uncertain planning factors is the amount of time and resources necessary for rework activities. More certainty about product quality therefore results in better project planning;
- For the same reason, the project lead-time is shortened.

These arguments sound good enough, but in practice seldom win from the

arguments given against improvement. However, there is a new development adding arguments in favour of improving. This important development is the increasing strategic use of software, i.e. supporting the primary business of an organisation, directly dealing with the customers and readily adaptable to new challenges. This means a different kind of customer than the traditional IT department, demanding quality software delivered in time. Improving only system and acceptance testing will result in failing to meet these "in time" demands. Improving developers' tests is an important step towards meeting the demands.

How to improve? An approach

Below, a practical approach on how to improve developers' tests is described. The approach has the following key aspects:

- organisation
- use of test design techniques
- use of a test life-cycle model

Organisation

Perhaps the most important aspect of the approach is that an **Application Integrator** (AI) will be made responsible for the progress of integration and for the quality of the outgoing product. The AI negotiates with the project manager or the development team leader what level of quality is required: under what conditions can the system be released to the next phase (exit-criteria). The AI also demands insight in the quality of incoming modules or programs (entry-criteria). A module or program is only accepted into the integration process if it meets the entry-criteria. In order to prevent mixing interests, the AI should not be development team leader. This creates a deliberate tension between the AI, who is responsible for quality, and the development team leader, who tends to focus on functionality and the amount of time and resources spent. Since the AI is a member of the development team, this generates far less resistance from the developers against improving their testing than other test approaches would do and considerably raises the awareness of quality within the development team. For test specific expertise, test professionals (outside of the project) support the AI. The AI communicates with the customer on quality issues. Because communication is essential, it is very important for the AI to have good social skills.

Use of test design techniques

The approach does not prescribe 100% use of formal test design techniques, as this will (in our experience) generate too much resistance. Instead, the AI, program testers and project manager or development team leader negotiate: important parts of the system will be tested using formal test design techniques, less important parts will be tested using informal techniques or even in the old-fashioned, undocumented way of testing without use of any techniques. A good balance has to be found, for which several aspects play a role:

- risks for the organisation / importance of the system;

- desired quality of the product;
- progress of the project;
- test maturity of the development team;
- test coverage;
- test evidence;
- resource consumption (of using the test design technique).

Popular techniques often applied in our projects are checklists and marking test situations in the functional specifications. Only test situations too complex to be tested directly from these markings are detailed further into specific test cases. Although definitely not watertight, the marked up documents supplied with the tester's initials serve as test evidence.

Use of a test life-cycle model

Exit- and entry-criteria and other agreements are laid down in a test plan (phase 1), test cases are prepared (phase 2) and executed (phase 3). Again, practical use prevails: writing documentation such as a test plan is not a target in itself, but serves as a means of communication between project leader, AI and customer, and should be kept as minimal as possible.

About the Speaker

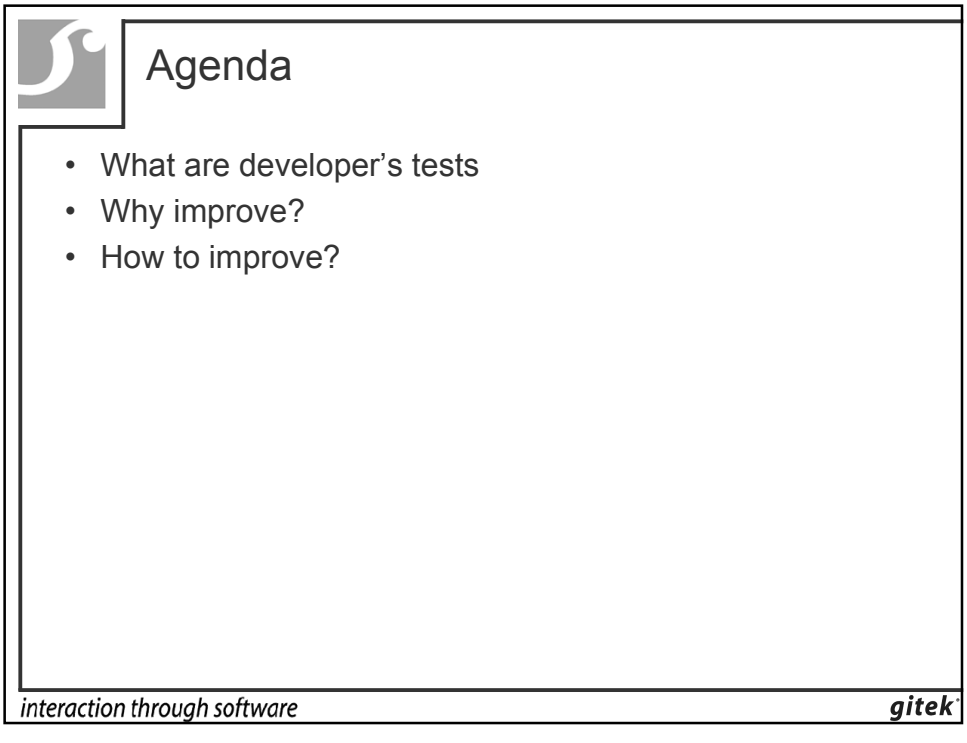
Since 1989, Ruud Teunissen is employed in the testing world. He has been involved in a large number of ICT projects and has performed several functions within the testing organization: tester, test specialist, test advisor, test manager, etc.

Based on his experience, Ruud participated in the development of the structured testing methodology TMap and is co-author of several books on structured testing. The last years Ruud is involved in implementing structured testing within organizations on the Belgian and Dutch market.

At this moment Ruud is working in Belgium for Gitek n.v. as Manager Testen. Ruud is frequently speaking in Benelux and Great Britain.

Improving developer's tests

Ruud Teunissen
Gitek n.v. *interaction through software*
http://www.gitek.be
e-mail: rt@gitek.be



Agenda

- What are developer's tests
- Why improve?
- How to improve?

Agenda

- ➲ What are developer's tests
- Why improve?
- How to improve?

Characteristics

Description

- Low-level tests: unit test, integration test, mostly performed by developers

Differences

- Defect finder = defect solver
- All defects should be repaired (no risk reporting)
- Attitude developer versus attitude tester
- Tests integral part of development process
- Most defects in product: fast & cheap repair

Agenda

- What are developer's tests
- ➲ Why improve?
- How to improve?

No improvement necessary!

- Good enough as it is
- Not enough time, too expensive
- A good (better) test follows!

"Not my job"

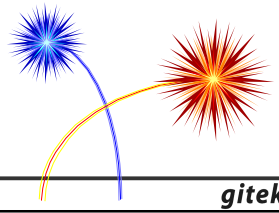
Benefits

Benefits to developer:

- Easier and cheaper repair
- Prevention of (similar) defects
- Less rework later
- Better planning
- Shortened lead time
- More certainty about product quality
- Better product quality

Benefits to total project:

1 + 1 = 3



Improvement is necessary

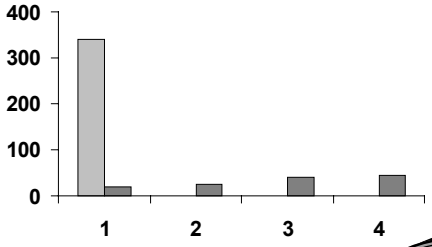
Customer demands:
QUALITY software
IN TIME!

Case

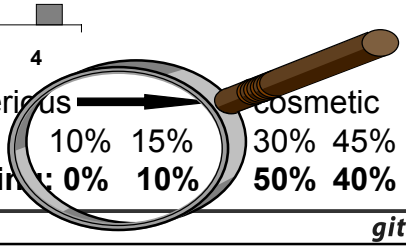
- Large bank
- Telebanking application
- Second try
- Clean start
- Team: 7 developers
- Duration: 8 months
- Size: 900 function points

Case results

- Defects from acceptance testing:
  - First try: 341 defects in 1 week
  - **Improved dev. testing: 130 defects in 4 weeks**



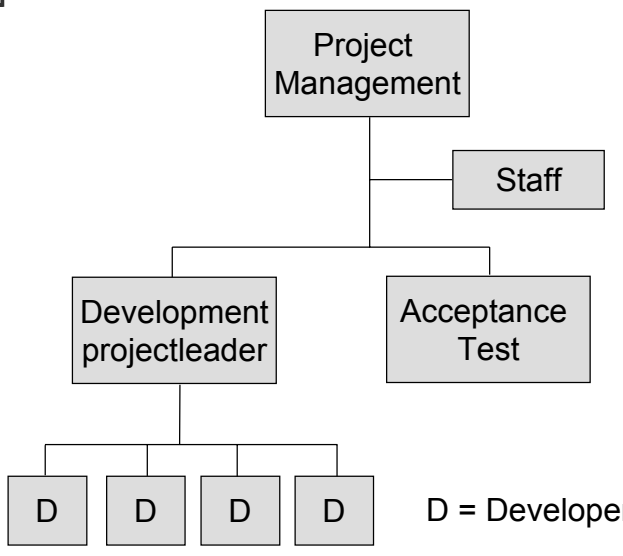
- Severity category: serious → cosmetic
  - First try: 10% 15% 30% 45%
  - **Improved dev. testing: 0% 10% 50% 40%**



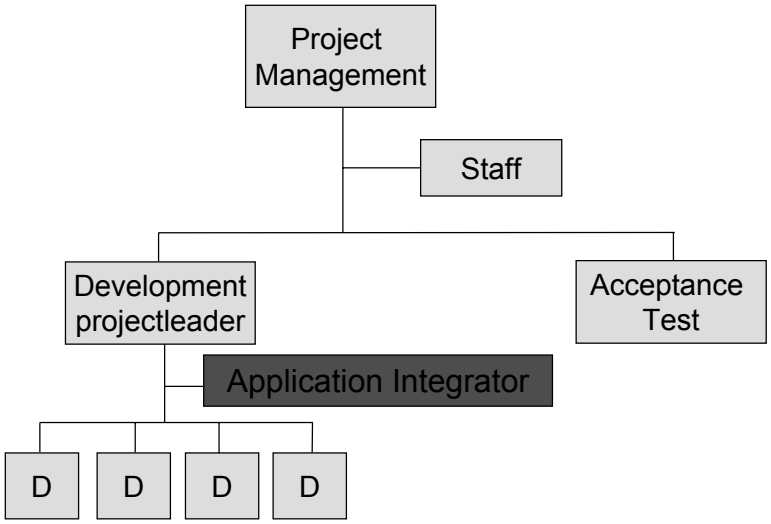
Agenda

- What are developer's tests
- Why improve?
- ➲ How to improve?

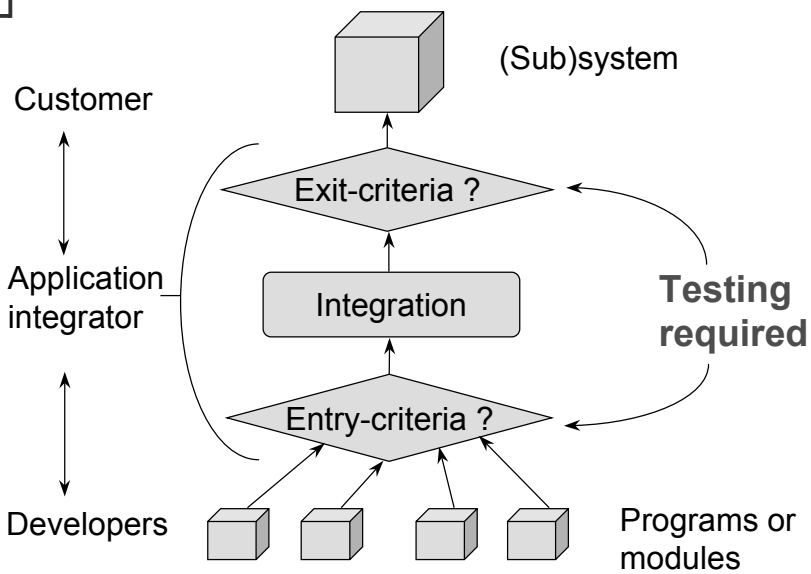
Traditional organisation



Role: Application integrator



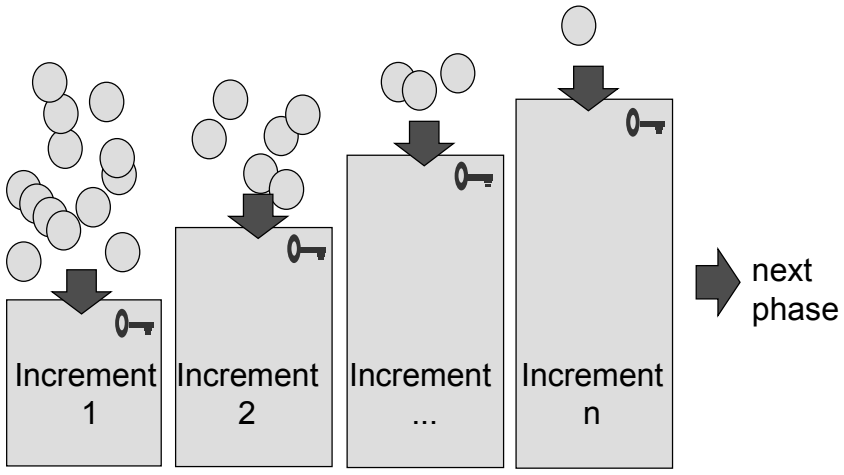
Quality and integration



Test strategy

- Be practical!
- “Right quality”
- Important functions:
  - better test coverage
  - more test evidence
- Important quality characteristics: the same
- Popular “techniques”:
  - Checklists
  - Marked up functional specifications
  - Only specification of test cases if necessary

Incremental integration

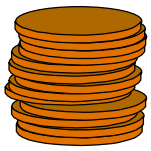


Use of a life-cycle model

- Test planning phase
  - agreements
  - strategy
  - ...
- Test specification phase
  - identify test situations, according to strategy
  - (if necessary) prepare test cases
  - ...
- Test execution phase
  - execute tests
  - analysis
  - reporting
  - conserve required evidence
  - ...

Summary

- Improving developers' tests is necessary:
  - Better product quality
  - Insight
  - Customer satisfaction & confidence
  - Quality awareness developers
- How to improve:
  - Application Integrator
  - Practical
  - Improvement from within development team, supported by professional testers



IT WORKS!!!

YOUR CHALLENGE!!


Questions?
Please ask !
or email:
rt@gitek.be
interaction through software
gitek

QWE2000 Session 5A

Dr. Erik P. vanVeenendaal

[Netherlands]

(Improve Quality Services BV) "GQM Based Inspection"

Key Points

- Inspections
- GQM
- Measurement programme
- Feedback sessions
- Practical results

Presentation Abstract

Inspections are generally accepted as a means to improve the quality of software products in an effective and efficient way. However, inspections are not a standard practice in a great number of software projects and software organisations. Introducing and implementing inspections is often a tedious and difficult task, because software engineers must be personally convinced of the effectiveness of new methods before they will consistently use them.

Collecting relevant data during inspections is a way to overcome these difficulties. Such data collection for software inspections is termed measurement. Measurement is a powerful aid to implement and improve the inspection process. Showing real-life data is often convincing for both the software engineers and their managers. A well-known and popular software measurement approach is the Goal/Question/Metric method (GQM). Applying GQM to the inspection process helps to focus the data gathering process, and support the interpretation process. An important part of the measurement programme and thus inspection implementation and improvement process are the so-called feedback sessions. Feed back sessions are meetings involving members of the project team and the measurement team. It is an essential tool for analysis and interpretation of the measurement results.

The background to this paper is the implementation of inspections in a number of Dutch organizations using the GQM approach as a main vehicle. Practical examples are provided of the measurement goals, metrics and feedback sessions.

About the Speaker

Dr. Erik P.W.M. van Veenendaal CISA has been working in as a practitioner and manager within the area of software quality for a great number of years carrying out assignments in the field of quality management, EDP-auditing, software testing and inspections. Within this area he specializes in software testing and is the author of a number of books, e.g. Structured testing; an introduction to TMap and Software quality from a business perspective.

As a test manager and consultant he has been involved in a great number and variety of projects, has implemented structured testing and carried out process improvements activities in a large number of organisations. He is a regular speaker both at national and international testing conferences and a leading international trainer in the field of software quality. Erik van Veenendaal is the founder and managing director of Improve Quality Services, a company that provides services in the area of quality management, testing and inspections.

At the Eindhoven University of Technology, Faculty of Technology Management, Erik is part-time involved in lecturing and research activities. He is on the Dutch standards institute committee for software quality.

GQM based Inspection

Erik van Veenendaal / Mark van der Zwan

Abstract

Inspections are generally accepted as a means to improve the quality of software products in an effective and efficient way. However, inspections are not a standard practice in a great number of software projects or organisations. Introducing and implementing inspections is often a tedious and difficult task, because software engineers must be personally convinced of the effectiveness of new methods before they will consistently use them.

Collecting relevant data during inspections is a way to overcome these difficulties. Such data collection for software inspections is termed measurement. Measurement is a powerful aid to implement and improve the inspection process. Showing real-life data is often convincing for both the software engineers and their managers. A well-known and popular software measurement approach is the Goal/Question/Metric method (GQM). Applying GQM to the inspection process helps to focus the data gathering process, and support the interpretation process. An important part of the measurement programme and thus inspection implementation and improvement process are the so-called feedback sessions. Feedback sessions are meetings involving members of the project team and the measurement team. It is an essential tool for analysis and interpretation of the measurement results.

The background to this paper is the implementation of inspections in a number of Dutch organizations using the GQM approach as a main vehicle. Practical examples are provided of the measurement goals, metrics and feedback sessions.

1. Software Inspections

In every software development phase defects are introduced, found and rework is being carried out. However, often most defects are only found when the software product is almost finished, e.g. during the system and acceptance testing phase, or even during operation. Defects found during the testing phase have the disadvantage that their rework on the almost finished software product is very time consuming. It would have saved the development organisation a lot of time if these defects where found during an earlier development phase.

Inspections are an effective and efficient quality technique that can be introduced to improve the quality of the products at an early stage [2]. Besides finding a defect at the earliest possible moment, the prevention of defects is the important issue. Inspections can also be used as a means for defect prevention. Based on an analysis of the defects that were found, the software development processes can be adapted and optimised to prevent these defects from occurring in the future (as far as possible). Engineers that are involved in the inspection process can learn from their defects or the defects that were made by someone else. Inspections can be defined as a structured review of an engineers' software work product carried out by his colleagues to find defects and to enable the engineer to improve the quality of the product [1].

While the importance and benefits of inspections for software projects is well understood within the software industry, only few engineers apply the inspection technique to their personal work. Even when statistic evidence from other organizations and projects exists [4] [7], the introduction of improved software methods, e.g. inspections, is often slow because software engineers must be personally convinced of the effectiveness of new methods before they will consistently use them. In software this is particularly true because [3]:

- software engineers' methods are largely private and not obvious from the products they will produce. Thus, if they do not use proper methods, it is unlikely that anyone else will know;
- software engineers are generally not trained to follow the planning and measurement disciplines needed to rigorously evaluate the methods they use;
- even when software groups have a common set of defined practices, these practices are not consistently followed;
- the current industrial environments do not as a prerequisite require the use of the best-known software engineering methods.

A principal issue, therefor, is how to motivate and implement inspections within a software project or organisation. The authors argue that metrics should play a major role in convincing both the software engineers and their management and tuning the inspection process. In fact metrics are a critical success factor to successful inspection implementation. Metrics are not optional, they are a requirement.

2. Goal/Question/Metrics approach

A well-known and popular software measurement approach is the Goal/Question/Metric approach [5] [6]. GQM represents a systematic approach to tailor and integrate goals with software process and products models. It is based on the specific needs of a project and organisation. Within GQM measurement goals are derived from high-level corporate goals, and further refined into measurable values (metrics). GQM defines a certain goal, refines this goal into questions, and defines metrics that must provide the information to answer these questions. The GQM paradigm provides a method for top-down metric definition and bottom-up data interpretation (figure 1).

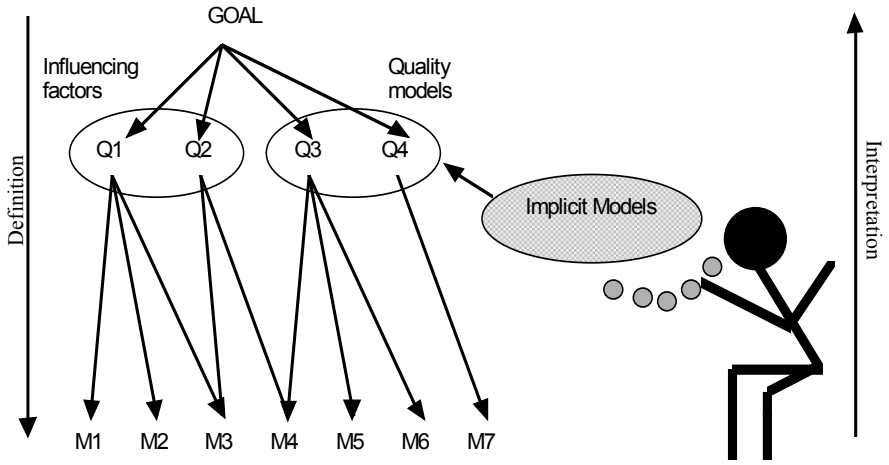


Figure 1 : The Goal Question Metric paradigm

A number of steps can be distinguished with the GQM process. The steps to take are applicable when introducing (inspection) measurement in practice. Figure 2 shows these steps.

Step 1: Organisation and project characterisation. Defining measurement programmes starts with a characterisation of the organisation and the project. The results of the characterisation are used in the definition, ranking and selection of the goals and also in establishing the GQM-plan.

Step 2: Goal definition. The second step in the GQM process is defining measurement goals. Goals can directly reflect business goals, but also specific project goals or personal goals. Measurement goals must be carefully selected, based on selection criteria such as: priority to the project or organisation, risk, time in which a goal can be reached.

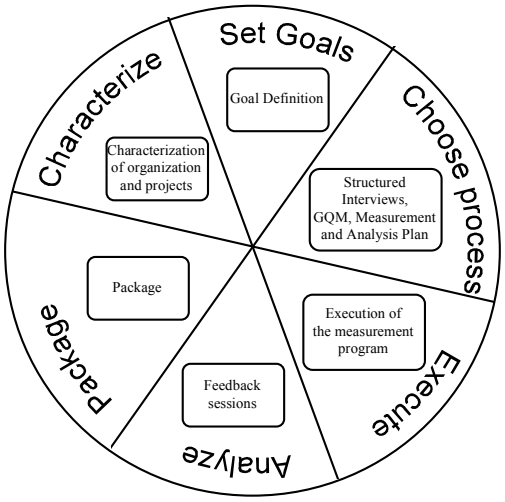


Figure 2 : The steps for goal-oriented measurement

Step 3: Developing the measurement programme. The major activity when developing a measurement programme is refining the selected goals into questions and metrics. It is important to check whether metrics answer questions, and that answers to such questions provide a contribution towards reaching the defined goals.

Step 4: Execution of the measurement programme. In the execution step of the measurement programme, the measurement data is collected according to the procedures defined in the measurement plan and the feedback material is prepared as described by the analysis plan.

Step 5: Feedback sessions. Feedback sessions are meetings involving members of the project team and the measurement team. It is an essential tool for analysis and interpretation of the measurement results A more detailed description of Feedback session is provided hereafter.

Step 6: Packaging of measurement results. To re-use measurement results and experiences, the results of the measurement programmes must be packaged. Packaging must be done in such a way that future projects, or other parties from the organisation, are able to use the measurement results.

Application of GQM measurement by means of the described process divides the improvement cycle in two parts. The first part consists of the definition process, during which goals, questions, metrics and additional procedures are defined (step 1, 2, and 3). The second part consists of the interpretation process, during which the collected data is being analysed, improvements are identified, and experiences are described (step 4, 5, and 6). For motivating and convincing software engineers the interpretation process, and especially the feedback sessions are the most important step.

3. Interpretation and feedback session

3.1 Interpretation process

Based on the gathered data, an analysis can be performed aimed at answering questions, and reaching goals. This is the "interpretation process". Research has shown that interpreting measurement data is a learning process within a software team. Interpreting measurement data in a solid way can be done according to the three principles of goal-oriented measurement:

- Software measurement must reflect the interest of the data providers and must be based on the knowledge of the development team.
- Only the software developers that provide the data, can interpret that data validity. They are the only ones who know all the details, also the ones that were *not* measured, and therefore are the only ones allowed to really interpret feedback material.
- Because of the limited amount of time of software developers (caused by their commitments to project planning), conflicts of interest may occur when the development team performs all measurement tasks. Therefore separate staffing must be available that support the collection and analysis of the measurement data, by performing all activities that do not necessarily have to be carried out by the development team. In the context of inspections these activities should be carried out by the inspection implementation team.

To get the most out of measurement, the interpretation must be emphasised to close the feedback loop. The main objective of software measurement is to evaluate current processes and identify improvement opportunities. Depending on the results immediate changes and adjustments on both the software development process and the measurement process can be suggested. Through defining conclusions and action points during the interpretation process, software process improvement is started at the project and engineering level. The motivation for software developers to participate in software measurement is mainly determined by the way the interpretation process is carried out. The most critical part of the interpretation process are feedback sessions.

3.2 Feedback sessions

Feedback sessions are meetings during which data is analysed by the development team based on the procedures defined in the GQM plan. These sessions are important to keep interest and motivation for the measurement programme. This is one of the reasons why they should be done often. On the other hand, there should be enough time between feedback sessions to ensure there is enough new measurement data and the effort spent is being optimised. This is a kind of paradox: on the one hand, feedback must be done often; on the other hand, this is not feasible. In practice, a balance is achieved by running feedback sessions every six to eight weeks, depending on the project specific goals. Generally feedback sessions last between two and three hours. A feedback session needs a high degree of concentration of the attendees. As an effect, a maximum number of 15 slides presenting measurement results can be discussed. Decisions have to made on the issues to discuss in a feedback session. In the first sessions of a measurement programme, it might be necessary to discuss all available material.

During a inspection feedback session also a changes to the inspection process can be given, new rules or checklists can be introduced. It may also be used to do a survey on the engineers' opinion regarding some aspect of inspection. At one organization feedback session were used to get quantitative data on the logging meeting by asking participant to score

statements such as “In the logging meeting we learn how to specify” and “In the logging meeting a common understanding reached”.

	Step:
1.	Freeze Database with measurement data
2.	Create Basic-set of analysis slides
3.	Create Additional-set with new analysis
4.	Select Feedback-material
5.	Distribute Feedback-material
6.	Analyse Feedback-material in the session
7.	Draw conclusions, answer questions, and define action points
8.	Evaluate with project team
9.	Report

Figure 3 : 9 Steps for Feedback Sessions

Feedback sessions consist of the steps shown in figure 3. The first step is to freeze the database in which the collected measurements are stored. With this data-set it is possible to update the basic set of slides to discuss during the feedback session. Extra slides can also be created to study the specific issues raised by the project team. For instance feedback was asked on average preparation, rework and throughput time. This data wasn’t present in the initial set of slides, but was requested by project members to improve their planning. Often the total set of slides is by then already too large. Therefore step 4 must be carried out, during which a selection is made on the subjects for the feedback session. This selection is done in co-operation with the representatives from the project. The slides are always preferably distributed to the attendees of a feedback session one or two days in advance, to offer the opportunity for preparation. Figure 4 shows an example of a slides from an inspection feedback session.

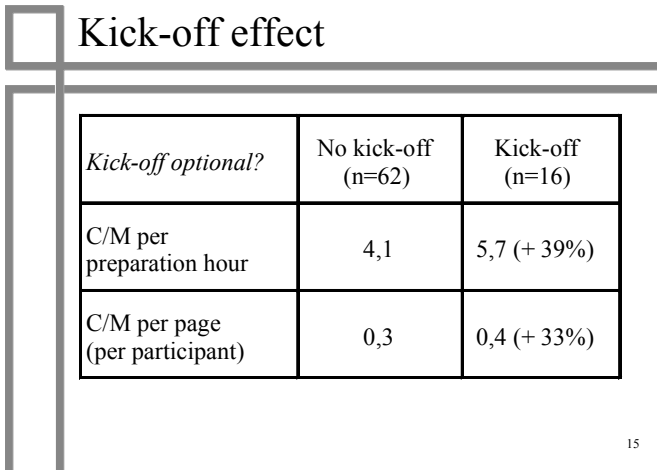


Figure 4 : Example of Inspection feedback session slide

After the individual preparations, the feedback session can take place. During a feedback session, a representative from the inspection implementation team guides the discussion.. He/she explains which data is included in a presentation slide, explains the axes of a chart, and if a relationship is visible from a slide, points at that relationship. After this explanation,

the development team is asked to interpret. Mostly the first interpretation already results in a group discussion, which is finished with an overall interpretation. At the end of the sessions a discussion takes place to identify two or three concrete inspection improvement points. The project team makes a group decision on the improvements to be made. The conclusions are recorded, and the action points are assigned among the participants of the feedback session. A feedback session is finalised by an evaluation with all participants involved. The results, conclusions and inspections improvement actions are reported by the inspection implementation team, including possible improvements for the measurement programme. The feedback process then comes to its end, but since it is a continuous process, measurement on inspection improvement actions are carried out and the next feedback session is already planned……

References

[1] Fagan, M.E. (1986), Advances in software inspections, in: IEEE Transactions on Software Engineering, vol. 12, no. 7, July 1986

[2] Gilb, T and D. Graham (1993), Software Inspections, Addison Wesley

[3] Humprey, W.S. (1995), A discipline for software engineering, Addison Wesley

[4] Rooijmans, J., H. Aerts and M. van Genuchten (1996), Software Quality in Consumer Electronic Products, in: IEEE Software, January 1996

[5] Solingen, R. van, and E.P.W.M. van Veenendaal (1997), Achieving software quality by GQM measurement, in: E. van Veenendaal and J. McMullan (eds.), Achieving Software Product Quality, Tutein Nolthenius, 's Hertogenbosch, The Netherlands

[6] Solingen, R. van, and E. Berghout (1999), The Goal Question Metrics method, McGrawHill

[7] Van Veenendaal, E.P.W.M (1999), Practical Quality Assurance for Embedded Software, in: Software Quality Professional, Vol. 1, no. 3, June 1999

The Authors

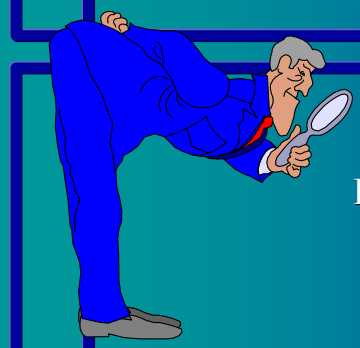
Erik van Veenendaal
Improve Quality Services BV
Eindhoven University of Technology
The Netherlands
e-mail : eve@improveqs.nl

Mark van der Zwan
Improve Quality Services BV
The Netherlands
e-mail : mzw@improveqs.nl

www.improveqs.nl

IMPROVE
Quality Services

GQM based Inspections

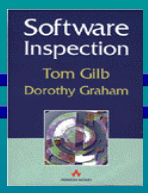


Do you have numbers ?
Watts Humphrey

Erik van Veenendaal
Mark van der Zwan

1

Inspections



- Structured peer review
- Initially developed at IBM by M.E. Fagan
  - enhanced by Tom Gilb / Dorothy Graham
- Objective:

  not just to find defects,

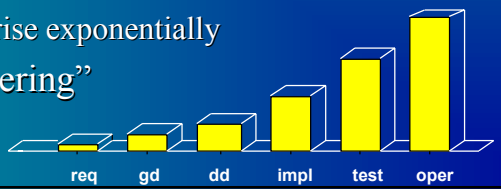
  but also,

  to find defects *earlier* in the life cycle and to remove the *causes* from the process

2

Why inspections ?

- 60% of the defects have already been made before coding/implementation has started
  (50% reported by *Freimut et al, 2000*)
  (51% "requirements related" reported by *Vinter, 1998*)
- Early defects are often the most important
  - defects have the characteristic to multiply themselves top-down
  - cost of rework rise exponentially
- "Egoless engineering"
  - Weinberg



3

The great contradition

- Lots of proof available
  - » 9.3 to 1 (Philips - UK), source: Tom Gilb
  - » 7.7 to 1 (Raytheon - USA), source: IEEE
  - » 2.1 to 1 (Philips Semiconductors - NL), source: SQP
  - » 13 to 1 (Datastream - USA), source: SQP
  - » ……..
- SEI Research 20% are doing inspections
- Personal experience: 10% as a maximum !!

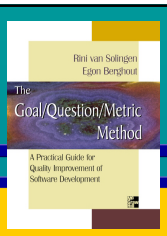
Do people really understand ??

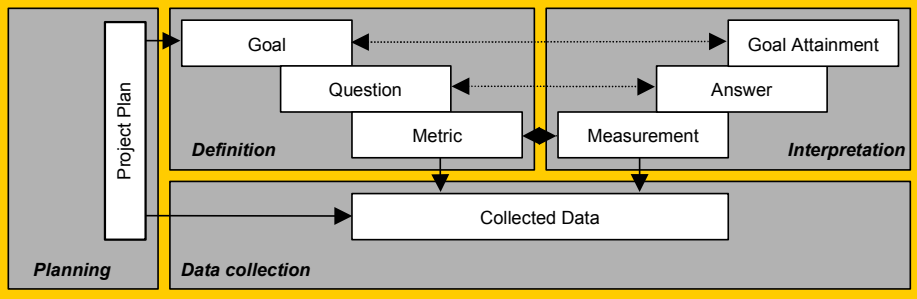
What to do ??

- Raise organizational and management awareness
- Train engineers on real inspections
- Pick some documents that really count
- …….
- Convince them by using their metrics

> "measurement must be focused, based upon goals and models" *Victor Basili*
>
> "easy to get 'numbers', what is hard is to know they are right and understand what they mean" *Bill Hetzel*

The GQM method



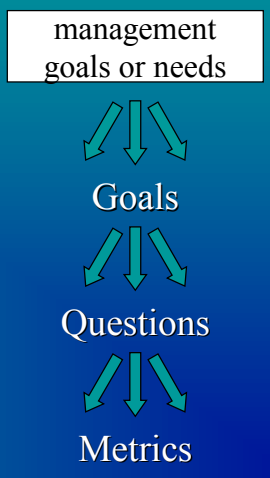


WWW.GQM.NL

Focusing: GQM

- Goal-Question-Metric

1. Development of goals
2. Generation of questions that define the goal
   - a list of questions that need answers in order to know whether the goals have been met
3. Identification of metrics that answer the question



7

Measurement (sub)goals

- Tune the review/inspection process
  - what are the influences factors ?
  - which criteria should be defined ?
- Insight in implementation status
  - are we doing inspections ?
  - how many engineers are trained ?
- Insight in the quality of the process
  - are we getting better ?
- Insight in (quantitative) results
  - how many defects have been found ?
  - what is the Return On Investment ?

8

Metric definitions

- Name & definition
  - » measurement formula's
  - » unambigious (as far as possible)
  - » explaining & examples
  - » e.g. what is a major defect?
- Hypothesis
  - » influencing factors
- Process form as result !!
- Data gathering procedures *(check the data)*
- Responsibilities

9

Common pitfalls

- Definition of major versus minor
  - definitions including defect type
- Measurement on number of pages
  - e.g. diagrams, LOC versus pages, document type
  - table of contents, front page etc.
- What is included in the various phases ?
  - e.g. preparation as one activity, rework

➨ Moderator sessions to discuss interpretation and refine definition

10

Interpretation: Feedback

- Feedback sessions
  - interpretation: discussion on metrics
  - present metrics not results !!
  - "QA doesn't know"
  - Beware your level 1……..
  - Let someone play the critic
- Create basic set
- Short feedback cycles (e.g. 10 inspections)
- Open to "everyone"

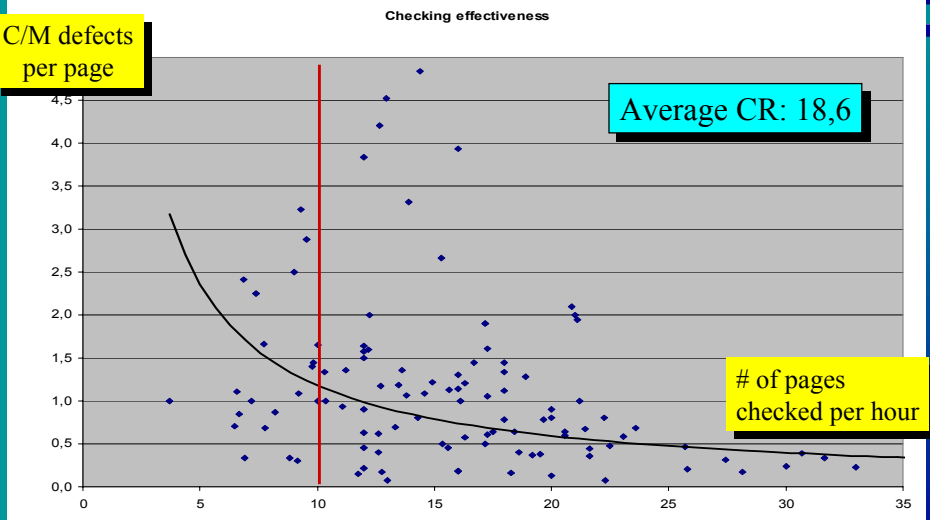
11

Tune the process - Kick-off effect

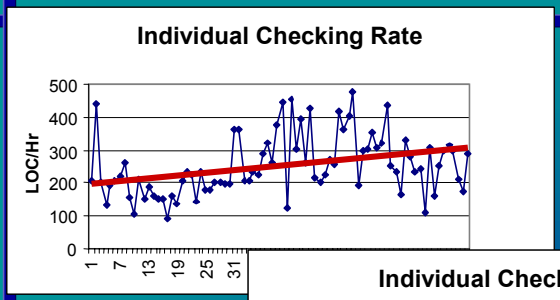
Kick-off effect	No kick-off (n=38)	Kick-off (n=107)
C/M per preparation hour	4,37	5,67 (+ 30%)
C/M per page (per participant)	0,26	0,46 (+ 73%)

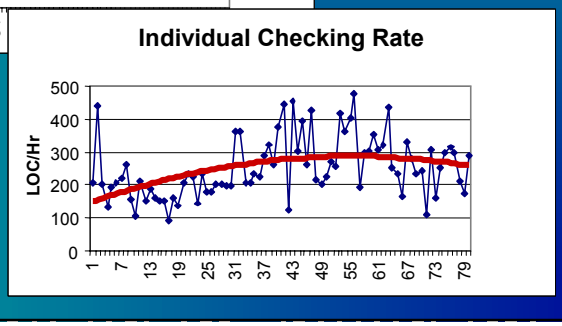
12

Tune the process - Checking rate

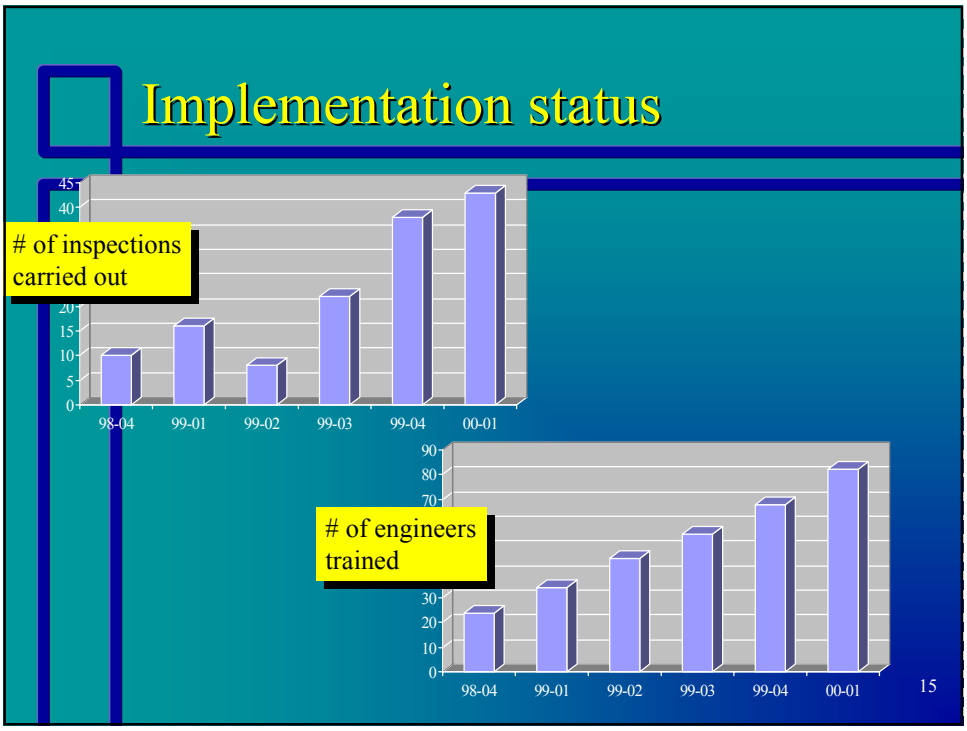


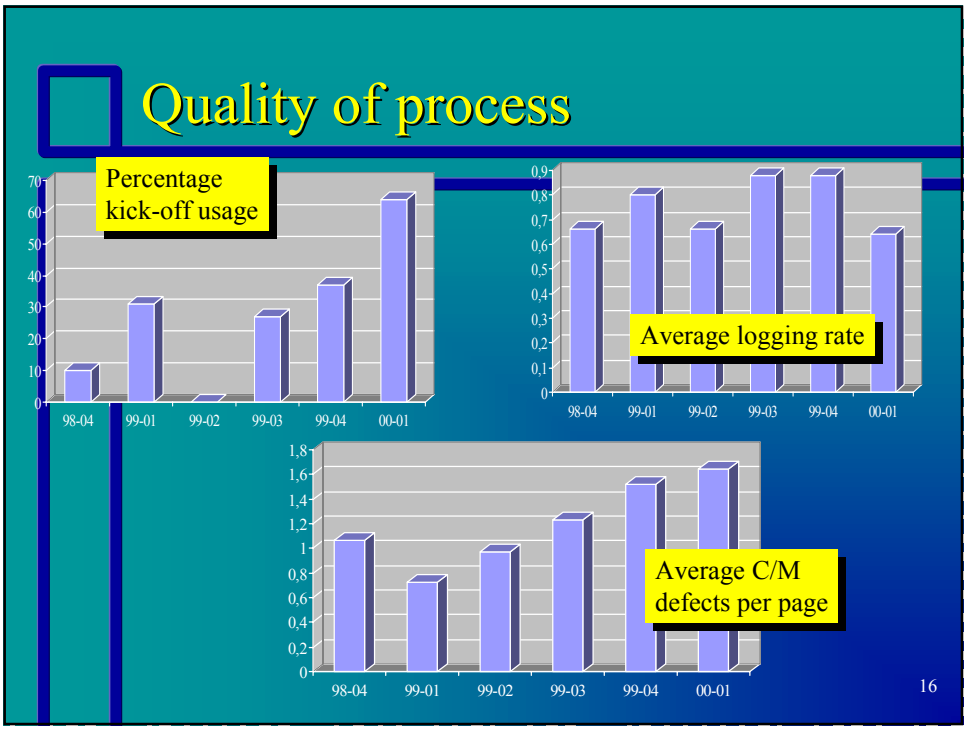
Another pitfall…….





14

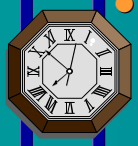




Results - Time saved

Field defects… !!??

- Number of inspection defects (Criticals) (X1)
- Number of inspection defects (Majors) (X2)
- Average rework time in testing (historical) (Y)
- Potential "additional" rework effort (X * Y)
- Time spent on inspections (Z)
  - including rework, follow-up & administration



- Time saved = ((X1 + (X2 x 0.3)) * Y) - Z
- ((467 + (1896 x 0,3)) * 2,75 h) - 2162 h = 686 h

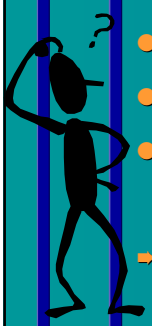
17

Other feedback aspects….

- Get additional information (engineers' opinion)
  - e.g. effect of the logging meeting
  - CMM compliancy (quick-scan)
- Changes on the process
  - new guideline
  - changes process form
  - etc...
- Agree on improvements
  - they choose !!

18

Results - Engineers' opinion

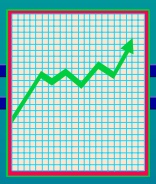


Quality of the products is improved	8,2
The software process is improved	7,6
Efficiency of development is improved	7,3
Learn how to specify	7,3
Our project is better controlled	7,2
A common understanding is reached	7,2

- also look at the standard deviation !!

19

Improvement possibilities



- Improve checking rate / logging rate
  - quantify
- Document inspection strategy
- Train new engineers
- Average number of pages
  - quantify
- Apply chuncking / sampling
- Carry out kick-off meetings
- .....

20

The GQM based approach

- PLAN
  - define process
- DO
  - JUST DO IT !!
- CHECK
  - GQM measurements
- ACT
  - bottom-up improvements

21

Summary pitfalls

How to screw up in spite of a good method

- Fail to collect data
- Failure to use data and give feedback
- When participants / moderators are not trained
  - amateurs will find ways to screw it up
- Everyone is a moderator
- Not following the rules, which usually work
  - until you know why (metrics !!) and how to modify them
- Not improving the inspection (and metrics!!) process using the ideas from participants

22

Finally ...

Inspections work !

It's easy but not simple !

Thank you

23

QWE2000 Session 5I




Mr. Rakesh Agarwal, Bhaskar Ghosh, Santanu Banerjee & Soumyendu Pal [India]

(Infosys Technologies Ltd)

"Challenges And Experiences In Establishing WebSite Quality (5I)"

Key Points

- Architecture
- Web Testing
- Simulatioin
- Quality Software

Presentation Abstract

Web technology has matured rapidly over the last few years. Previous web benchmarking focused merely on the number of ôhitsö that a web site could handle within any given time period. Now that the Internet is mainstream, and literally millions of people utilize the technology, organizations need more meaningful metrics to accurately evaluate technology implementation Businesses looking to capitalize on Internet technology require benchmarks that provide statistics based upon realistic end-user experiences. Statistics of importance include: average wait time for a dynamically generated HTML page to be delivered to an end user under heavy server loads; scalability of a provided web application solution when additional servers are incorporated into an existing server cluster; and maintenance support if a catastrophic server failure occurs. Website testing has much in common with the testing of standard

client/server applications. However, there are unique considerations that can affect the focus of testing strategy. In this paper we will discuss some of the factors that determine what to test in a Web site, the special considerations of database-driven Web sites, and how new Web test tools can help.

About the Speaker

Rakesh Agarwal is working with Infosys Technologies Limited, India, in the Education and Research Department for the past 3 years. He has published more than 60 papers in leading conferences and Journals.

Bhaskar Ghosh is working as Associate Vice President in Infosys Technologies Limited, India for the past 4 years. He has lead many projects in his carrier and Heads one of the Development Centers of Infosys.

Santanu Banerjee is a project leaders at Infosys Technologies Limited, India. He has been working on various web projects and currently involved in the design and development of a complete investment portal.

Soumyendu Kishore Pal is working in Infosys Technologies Limited, India. He has worked on number of leading projects and has interest in Web based testing.

Challenges and Experiences in establishing WebSite Quality

Rakesh Agarwal, Bhaskar Ghosh, Santanu Banerjee and Soumyendu Kishore Pal
Infosys Technologies Ltd., Near Planetarium, N.H.5,
Bhubaneswar - 751013, India, rakesh_a@inf.com

Abstract
Web technology has matured rapidly over the last few years. Previous web benchmarking focused merely on the number of "hits" that a web site could handle within any given time period. Now that the Internet is mainstream, and literally millions of people utilize the technology, organizations need more meaningful metrics to accurately evaluate technology implementation Businesses looking to capitalize on Internet technology require benchmarks that provide statistics based upon realistic end-user experiences. Statistics of importance include: average wait time for a dynamically generated HTML page to be delivered to an end user under heavy server loads; scalability of a provided web application solution when additional servers are incorporated into an existing server cluster; and maintenance support if a catastrophic server failure occurs.

Website testing has much in common with the testing of standard client/server applications. However, there are unique considerations that can affect the focus of testing strategy. In this paper we will discuss some of the factors that determine what to test in a Web site, the special considerations of database-driven Web sites, and how new Web test tools can help.

1. Introduction

The Internet has raised consumers' expectations to new highs. Whether you sell goods or services, alternatives are likely to be only a click away. The users are not willing to wait more than a few seconds for pages to come when they're clicking around your site. How do we make sure your Web site delivers top performance during peak demand periods? How do we build in the flexibility to deal with a sudden spike in traffic when we launch a new product? How do you handle an attack from vandals intent on flooding out the legitimate traffic?

This answer is high availability: building your Web site infrastructure such that regardless of an assault, no single part ever gets so overloaded that it fails or slows the site to a crawl. The basic elements of a high-availability Web site are equipment, connections, and skill[1][2].

Within minutes of going live, a WWW application can have many thousands more users than a conventional, non-WWW application[3][4]. The immediacy of the WWW creates an immediate expectation of quality and rapid application delivery, but the technical complexities of a WebSite and variances in the browser make testing and quality control more difficult, and in some ways, more subtle. Testing of WebSites is both an opportunity and a challenge. A WebSite can be complex, and that complexity -- which is what provides the power, of course -- can be an impediment in assuring WebSite

Quality[5][6]. Some of the major components of WebSites as seen from a Quality perspective are:

Browser: The browser is the viewer of a WebSite and there are so many different browsers and browser options that a well-done WebSite is probably designed [7][1][6]to look good on as many browsers as possible. This imposes a kind of de facto standard: the WebSite must use only those constructs that work with the majority of browsers. But this still leaves room for a lot of creativity, and a range of technical difficulties.

Database Access: In E-commerce, applications[8][9] either the data is build up or retrieve from a database. How does that interaction perform in real world use? If a "correct" or "specified" input is given does the result produce the expected output?
Some access to information from the database may be appropriate depending on the application, but this is typically found by other means.

Multi-Media: What about streaming video, audio, online chats? How is quality assessed here? What are the validation mechanisms? How do we know if the presentation is right?

Navigation. Users move to and from pages, click on links, click on images (thumbnails), etc. Navigation in a WebSite often is complex and has to be quick and error free.

Object Mode: The display changes dynamically; the only constants are the "objects" that make up the display. These aren't real objects in the Object Oriented sense; but they have to be treated that way. So, the quality test tools have to be able to handle URL links, forms, tables, anchors, buttons of all types in an "object like" manner so that validations are independent of representation.

Server Response: How fast the WebSite host responds influences whether a user (i.e. someone on the browser) moves on or continues. Obviously, InterNet loading affects this too, but this factor is often outside the Webmaster's control at least in terms of how the WebSite is written. Instead, it seems to be more an issue of server hardware capacity and throughput. Yet, if a WebSite becomes very popular loading and tuning are real issues.

Interaction and Feedback: For passive content-only sites the only issue is availability, but for a WebSite that interacts with the user, how fast and how reliable that interaction is can be a big factor.

Concurrent Users: Do multiple users interact on a WebSite? Can they get in each others' way? While WebSites often resemble client/server structures, with multiple users at multiple locations a WebSite can be much different, and much more complex, than complex applications. WebSite's quality and reliability [10][1][11]are crucial. The very special nature of the WWW and WebSites pose unique software testing challenges. Webmasters, WWW applications developers, and WebSite quality assurance manages need tools and methods that can match up to the new needs[12[13]. Mechanized testing via special purpose WWW testing software offers the potential to meet these challenges.

2. Process for testing the WebSite

There are certain procedural steps that need to be followed in order to avail a good WebSite that adheres to the standards[14[15] of any quality system [1][6]. The steps in website testing are:

- Define the purpose of WebSite testing effort.
- Develop test plan/scenarios.
- Running and evaluating the test plans/scenarios.
- Continuos testing and measurement.

Figure 1 shows the process for testing the WebSites.

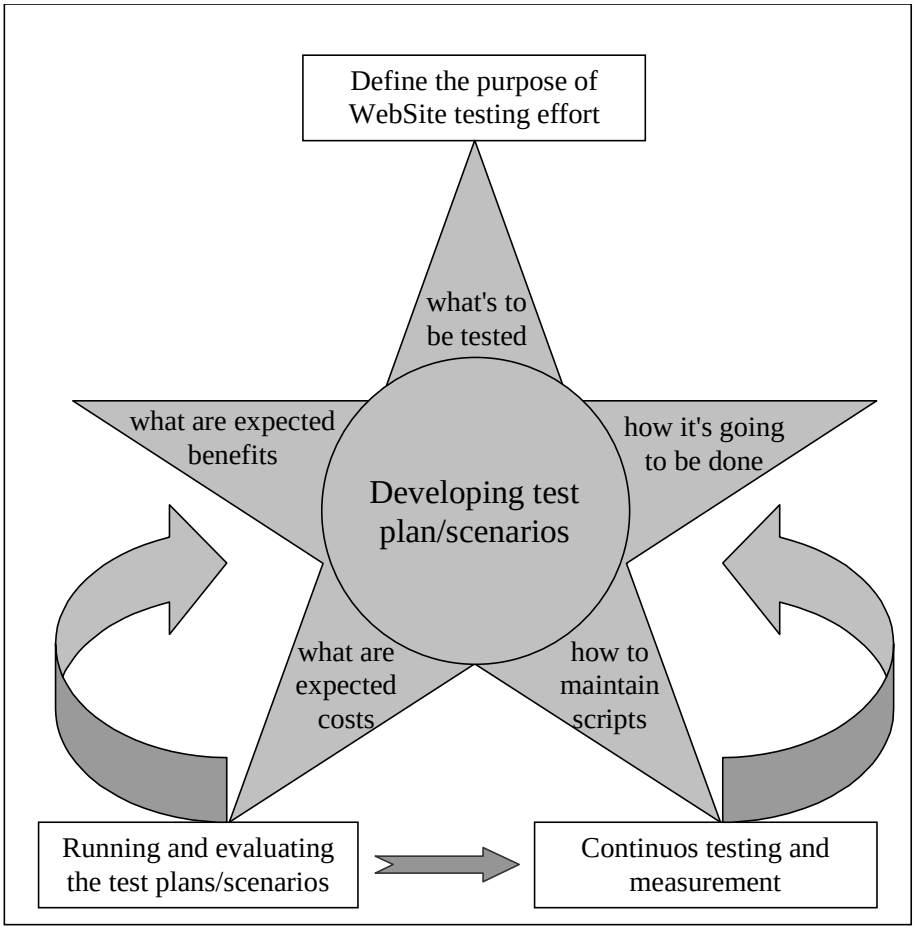


Figure 1: Phases in testing WebSite

2.1 Define the purpose of WebSite testing

There are several categories of testing each with its own purpose[3][16]. Identifying what needs to be tested. A WebSite is like any piece of software: no single quality measure applies, and even multiple quality metrics may not apply. Yet, verifying user-critical impressions of "quality" and "reliability" is a big task for WebSites. There are many

dimensions of quality, and each measure will pertain to a particular WebSite in varying degrees.
Time: WebSites change often and rapidly? How much has a WebSite changed since the last upgrade? How do we highlight the components that have been changed?

Structural: How well do all of the parts of the WebSite hold together. Are all links inside and outside the WebSite working? Do all of the images work? Are there parts of the WebSite that are not connected?

Content: Does the content of critical pages match what is supposed to be there? Do key phrases exist continually in highly-changeable pages? Do critical pages maintain quality content from version to version? What about dynamically generated HTML pages?

Accuracy and Consistency: Are today's copies of the pages downloaded the same as yesterday's? Is it close enough? Is the data presented accurate enough?

2.2 Develop test plan/scenarios.

This is very important in mapping out what's to be tested[7][8], how it's going to be done, how the scripts will be maintained and what the expected costs and benefits. Structurally every testing effort should have a testing strategy, or test plan, so should there be a *plan* built for WebSite testing.

Response Time and Latency: Does the WebSite server respond to a browser request within certain parameters? In an E-commerce context, how is the end-to-end response time after a SUBMIT button is pressed? Are there parts of a site that are so slow the user declines to continue working on it?

Performance: Is the Browser-Web-WebSite-Web-Browser connection quick enough? How does the performance vary by time of day, by load and usage? Is performance adequate for E-commerce applications? Taking 10 minutes to respond to an E-commerce purchase is clearly not acceptable!

Impact of Quality. Quality is in the mind of the user. A poor-quality WebSite, one with many broken pages and faulty images, with CGI-Bin error messages, etc. may cost in poor customer relations, lost corporate image, and even in lost sales revenue. Very complex WebSites can sometimes overload the user.

The combination of WebSite complexity and low quality is potentially lethal. Unhappy users will quickly depart for a different site! And they won't leave with any good impressions

2.3 Running and evaluating the test plans/scenarios.

The entire process of testing should be treated as a software development effort[3][4]6]. This includes defining what should be automated, (the requirements phase), designing test automation, writing the scripts, testing the scripts, etc. The scripts need to be maintained over the life of the product just as any program would require maintenance.

Other components of software development[17][18], such as configuration management also apply. By simulating the test plan in iterative manner, we will get the quantitative outputs.

2.4 Continuos testing and measurement.

The effort of testing is an investment. Sufficient time and resources are needed in order to obtain the benefits. Definitely, some scripts can be created which will provide immediate payoff, but this is not much of benefit. The benefit comes from running these automated tests every subsequent release[19][20]. Therefore, ensuring that the scripts can be easily maintained becomes very important[7].

3. Assuring WebSite quality

Assuring WebSite quality requires conducting sets of tests, automatically and repeatable, that demonstrate required properties and behaviors. Some required elements of tools that aim to do this are:

Browser Independent: Tests should be realistic, but not be dependent on a particular browser, whose biases and characteristics might mask a WebSite's problems.

No Buffering, Caching: Local caching and buffering -- often a way to improve apparent performance -- should be disabled so that timed experiments are a true measure of the Browser-Web-WebSite-Web-Browser response time.

Fonts and Preferences: Most browsers support a wide range of fonts and presentation preferences, and these should not affect how quality on a WebSite is assessed or assured.

Object Mode: Edit fields, push buttons, radio buttons, check boxes, etc. All should be treatable in object mode, i.e. independent of the fonts and preferences. Object mode operation is essential to protect an investment in tests and to assure tests' continued operation when WebSite pages change. When buttons and form entries change location -- as they often do -- the tests should still work.
When a button or other object is/are deleted, that error should be sensed! Adding objects to a page clearly implies re-making the test.

Tables and Forms: Even when the layout of a table or form varies in the browser's view, tests of it should continue independent of these factors.

Frames: Windows with multiple frames ought to be processed simply, i.e. as if they were multiple single-page frames.

Test Context. Tests need to operate from the browser level for two reasons: (1) this is where users see a WebSite, so tests based in browser operation are the most realistic; and (2) tests based in browsers can be run locally or across the Web equally well. Local execution is fine for quality control, but not for performance measurement work, where response time including Web-variable delays reflective of real-world usage is essential.

4. Website validation processes

Confirming validity of what is tested is the key to assuring WebSite quality -- and is the most difficult challenge of all. There are five essential areas where test automation will have a significant impact.

4.1 Operational Testing

Individual test steps may involve a variety of checks on individual pages in the WebSite:

- *Page Quality:* Is the entire page identical with a prior version? Are key parts of the text the same or different?
- *Table, Form Quality*: Are all of the parts of a table or form present? Correctly laid out? Can it be confirmed that selected texts are in the "right place".
- *Page Relationships:* Are all of the links a page mentions the same as before? Are there new or missing links?
- *Performance, Response Times*: Is the response time for a user action the same as it was (within a range)?

4.2 Test Suites

Typically there are dozens or hundreds (or thousands?) of tests, and we may wish to run tests in a variety of modes:

- *Unattended Testing:* Individual and/or groups of tests should be executable singly or in parallel from one or many workstations.
- *Background Testing*: Tests should be executable from multiple browsers running "in the background" [on an appropriately equipped workstation].
- *Distributed Testing*: Independent parts of a test suite should be executable from separate workstations without conflict.
- *Performance Testing*: Timing in performance tests should be resolved to 1 millisecond levels; this gives a strong basis for averaging data.
- *Random Testing:* There should be a capability for randomizing certain parts of tests.
- *Error Recovery:* While browser failure due to user inputs is rare, test suites should have the capability of re-synchronizing after an error.

4.3 Content Validation

Apart from how a WebSite responds dynamically, the content should be checkable either exactly or approximately. Some of the ways are:

- *Structural:* All of the links and anchors match with prior "baseline" data. Images should be characterizable by byte-count and/or file type or other file properties.
- *Checkpoints, Exact Reproduction*: One or more text elements -- or even all text elements -- in a page should be markable as "required to match".
- *Gross Statistics:* Page statistics (e.g. line, word, byte-count, checksum, etc.).
- *Selected Images/Fragments:* The tester should have the option to rubber band sections of an image and require that the selection image match later during a

subsequent rendition of it. This ought to be possible for several images or image fragments.

4.4 Load Simulation

Load analysis needs to proceed by having a special purpose browser act like a human user. This assures that the performance checking experiment indicates true performance -- not performance on simulated but unrealistic conditions.

Sessions should be recorded live or edited from live recordings to assure faithful timing. There should be adjustable speed up and slow down ratios and intervals. Load generation should proceed from:

- *Single Browser:* One session played on a browser with one or multiple responses. Timing data should be put in a file for separate analysis.
- *Multiple Independent Browsers*: Multiple sessions played on multiple browsers with one or multiple responses. Timing data should be put in a file for separate analysis. Multivariate statistical methods may be needed for a complex but general performance model.
- *Multiple Coordinated Browsers:* This is the most-complex form -- two or more browsers behaving in a coordinated fashion. Special synchronization and control capabilities have to be available to support this.

5. How do you impose load on a website ?

The main goal of creating an artificial load on a WebSite is to permit the load to emulate one or a dozen or a hundred or thousands of users actually using the WebSite. There are two main ways to do this. Caveats: First one on the Web, this may be quite difficult. Other on the LAN this is easier, but may require a 100Mbps LAN to saturate the servers. Hope: At some point the capacity will scale linearly: 2X machines means 2X capacity.

HTTP Protocol Based -- No matter what a user does on a WebSite the HTTP protocol prevails, so it is natural enough to start thinking about imposing load by finding a way to simulate the HTTP protocol. You can relatively easily create or record HTTP requests by a browser to the candidate WebSite. There are many tools that can do the get URL page [a simple command/action] -- and do this under user control. A simple enough can be a PERL script.

While relatively easy to use to generate basic retrievals of pages, this approach suffers from the fact that all of the URLs associate with a page have to be included if the simulation is to be a realistic one. The disadvantage is that you could easily create a test scenario that fails to include important load factors such as download times for images and other slow-to-respond page components.

Browser Based Playback Simulations --. In this approach, the test scenarios are scripts that control a test browser and, working through the browser, emulate the typical users actions, responses and timings. This method has the advantage of reality, but may involve

more work in deciding on a scenario and making sure that the scenario plays back realistically.

6. Web Capacity Testing

We all have had the exasperating experience of waiting too long for a page to arrive at our Web browser. Ultimately, if the response time is too long, we "click away" and do something else.

Even when the Web is heavily saturated with requests, if you are patient enough every page you request will -- ultimately -- be delivered to your browser. But that's not good enough. Too slow response times turn users away, or, worse yet, because the user has moved on to another page or context, important session data could be lost.

How slow is "too slow"? In other words, the Web Site sever is configured properly and effectively when it has "enough capacity to meet the customer demands".

Good engineering practice suggests that Web server machines have a Safety Factor of 2 or more. This means, that, when serving the design load, the Web servers ought to be running at about 50% of maximum capacity.

Another way to look at this is to look for the reverse information: when is the server delivering a request too slowly? While you may never really know what the peak deliverable capacity of the server is, you might be able to tell fairly accurately when the imposed server load (i.e. the queue of incoming requests) is not being served within a specified time limit, say 10 seconds. Then you surely will know when there is NOT enough capacity -- when this limit is exceeded.

But if the WebSite involves a two-tier or three-tier structure -- which is increasingly common for e-commerce sites -- then measuring one machine may lead to false conclusions. For example, all three machines in a three-tier structure could be achieving their performance goals -- that is, not setting off alarms -- but the overall cummulative application could still be "too slow" to satisfy users.

6.1 Can we measure response time from a typical users' perspective?

It may be simpler to measure response time at the client side of the picture, i.e. from the browser. If you can pre-establish a multi-case scenario that is typical of what users do, then it is possible to engineer series of experiments that measure WebSite performance against that specific scenario.

Even though it is relatively easy to measure if such tests run slower than a threshold -- for example, the 10 second overall response time limit -- this approach has a fundamental limit: it is only as good as the set of scenarios that you develop to emulate actual WebSite use.

6.2 What Affects WebSite Performance?

Many factors affect how fast a WebSite appears to a user. There are many, many stages between a RETURN typed on a browser and a completed [or complete enough] page being rendered on the client.

1. Client Machine Speed and LAN Factors. Before the request is delivered to the Web by the client machine it may have to work its way through the local LAN and enter the Web, where the connect speed and local saturation affects performance.

2. The Outbound Request to the Server - Before the request gets to the server the Web routing and other technical overhead produces delays.

3. The Server Response Time -- After the request gets to the server the response speed is affected by the current request backlog, where the request is in a multi-layer queue, and how long-running the last-tier request (typically some kind of database access) takes.

4. The Inbound Delivery to the Client. After the response is delivered by the server the reverse of the above sequence takes over: the delivered pages have to wend their way through the Web back to the client.

5. The Client Response Time. After the data arrives at the client access point client machine speed and local LAN factors come into play again.

6. The Client Rendering. The browser now has all the parts it needs; it is a matter of how long it takes the browser to render the page/image -- or at least a minimal part of it -- so that the user can take action

Note: An apology is due for the simplifications we assumed here here. In fact, the sequence is rather more complex because all of the machines along the way between a request by a browser and a response seen in the browser involve, typically, multi-programming, multi-threaded executions that dynamically adapt to changing Web conditions that are at least in part adjusting to the actual requests that are being discussed. Left out are such technologies as threading requests via different routs, packet re-transmits and asynchronous arrivals, and much LAN protocol complexity.

Still, the bottom line is clear: elapsed time perceived by the user actually occurs, as they say, in true real time.

7. Case study using the WebLoad tool

WebLoad tests[21] Web applications by generating Virtual Clients that simulate real-world loads. Users create JavaScript-based test scripts that define the behavior of the Virtual Clients and WebLoad executes these test scripts monitoring the application response graphically and statistically, and presenting the test results in real-time.

WebLoad incorporates functional testing into the scalability testing process allowing to accurately verify the scalability and integrity of Web applications at the per-client, per-transaction, per-instance level under defined load conditions. WebLoad saves the test results including data from the Load Generators (described in the paragraphs that follow) and the hosting hardware's performance monitor. We can view all or part of the data in real time or after the test Session is complete in tabular format or in graphical format.

The following diagram illustrates the configuration for a typical WebLoad Load Session. Each component in the Load Session is labeled (1-4).

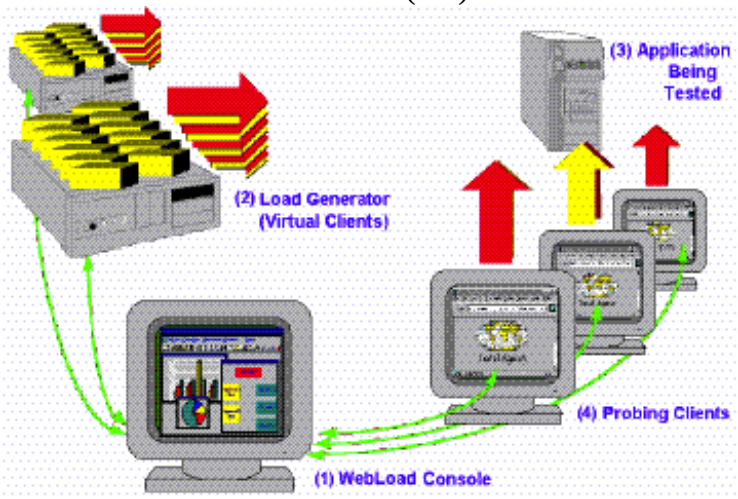


Figure 2: Configuration of typical Webload Load Session

- The Console sets up, runs and controls a test Session. At the Console, we can:
  - Define the hosts participating in the Load Session
  - Specify a program that the Load Session executes
  - Schedule tests
- The Load Machines are hosts, which run Load Generator software simulating multiple Virtual Clients. One Load Machine can run multiple Load Generators. Load Generators execute tests that "bombard" the application being tested with a large load, to enable complete scalability and stress testing. The tests consist of multiple simultaneous HTTP protocol requests. These requests are made from Virtual Clients (which emulate Web browsers) to Web servers. The Load Machines can run multiple threads and testing returns average values.
- The Application Being Tested is where the Web application being tested resides. The Application Being Tested (ABT) does not require that WebLoad software be installed on it.
- The Probing Client Machines are also hosts running Probing Client software, acting as a single Virtual Client and run at the same time as Load Machines – to further measure the performance of the Application Being Tested. WebLoad generates exact values for Probing Client performance. WebLoad uses a network agent, TestTalk to facilitate communication between the Console and hosts – either Load Machines or Probing Client Machines. TestTalk must be installed on both the Console and hosts.

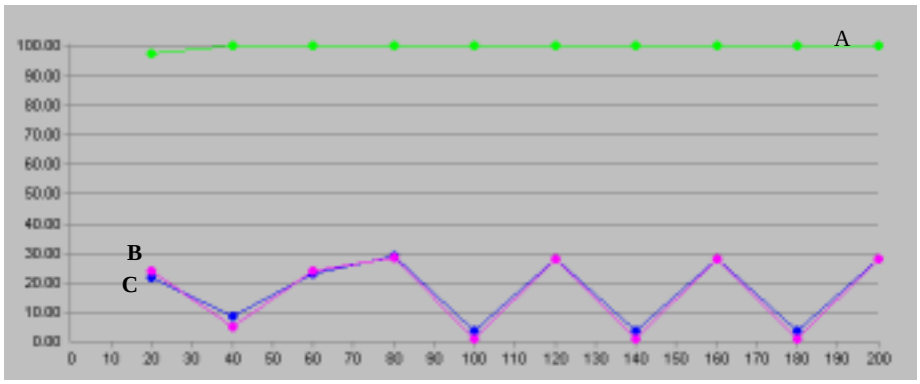
We have used the tool in order to test some of the projects. Some of the results are illustrated below:

------ Load Size (A)
------ Transactions per second (B)
------ Throughput(Bytes per second) (C)

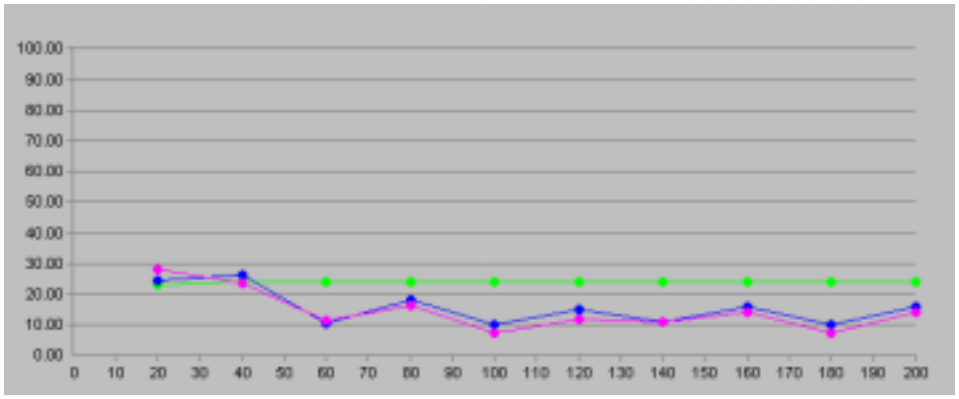
X: Axis – Session running time.
Y-Axis - Value in percentage

7.1 STRESS TESTING

7.1.1 12 Users simultaneously logged in

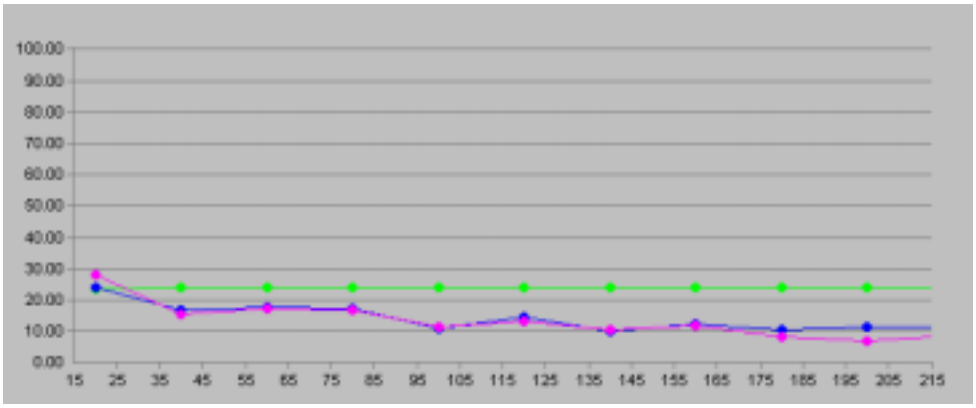


7.1.2 1 User logged in.

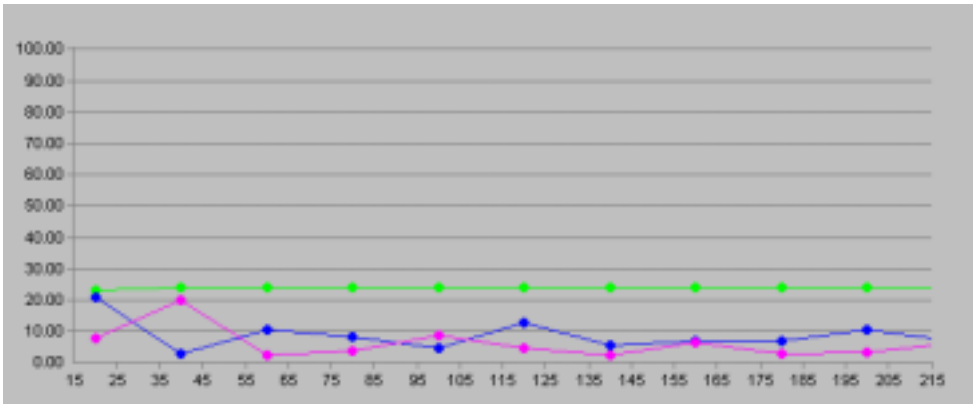


7.2 Connectivity with 12 Users logged in –

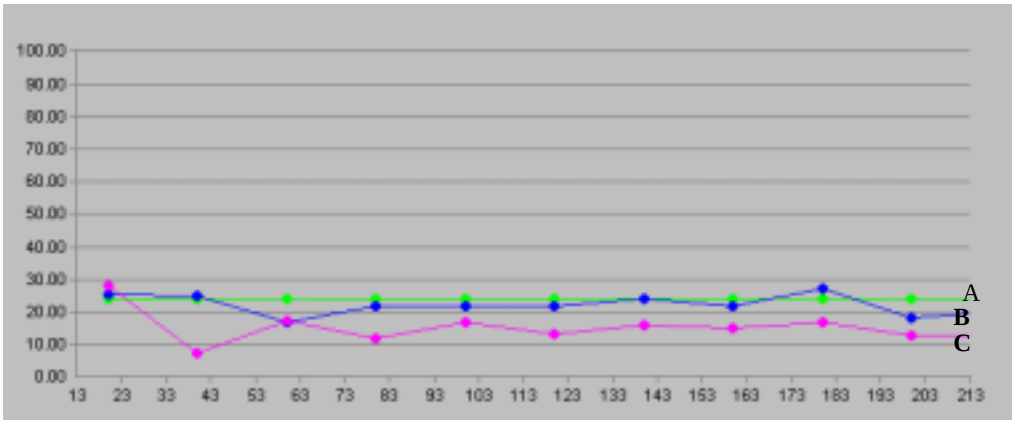
7.2.1 1.5 MBPS link



7.2.2 56kbps



7.3 Performance with 12 users with no cache…..



8. Observations from the project

We learned several valuable experience during the web based testing:

- Before building a first web-based application system, a pilot project (or an experimental project) must be implemented to get the valuable experience and lessons from the first practice. Through this pilot project, we can find various issues, alternative solutions to these problems,
- A team is needed to perform different feasibility and experimental study, high-level as well low-level designs, and implement some reusable and important components. Their major tasks are: 1) to identify the problems (including potential problems), 2) to find good and appropriate mechanisms (or solutions) to these problems, 3) to set up well-defined interfaces between components/sub-systems as well as client/server interface, 4) to create a set of good design patterns and implementation models, 5) to define well-structured code templates.
- Hypertext designs and implementations for a data-centered application system should be designed based on database schema and its data hierarchy structure.
- To simplify database accesses and the related communications between clients and the server, each database related hypertext form/template should be self-contained and consists of necessary data items, which can be easily used to perform database transactions and queries.

The weakness of most web methodologies lies in their approach. But testing large WebSite requires an approach. We presented a novel approach for testing WebSites. The proposed ideas are the result of many years of research mainly in the area of enterprise modeling[6[20]. They are currently being implemented and tested in a modeling environment. They proved to be very effective in handling complex models, in particular modeling a number of business processes. Further study will be devoted to operational issues deriving from the real use of tools. Such issues will provide the basis for new extensions to mainstream testing methodologies[18].

This is just a beginning of the battle for the WebSite testing. Many vendors are updating the existing architecture and some more new architectures possible[17][18]. Taking this work as base one may study about new extensions or easily understand the new architecture[17][18].

9. References

[1] James A. Whittaker, “What is Software Testing? And Why is it so hard”, IEEE Software, January/February 2000, pp. 70-79.

[2] Jerry Gao, Cris Chen, Yasufumi Toyoshima, and David Leung Developing an Integrated Testing Environment Using the World Wide Web Technology in Proceedings of the COMPSAC '97 - 21st International Computer Software and Applications Conference, 1997.

[3] Ji-Tzay Yang, Jiun-Long Huang, Feng-Jian Wang William. C. Chu. An Object-Oriented Architecture Supporting Web Application Testing in the Proceedings of

the Twenty-Third Annual International Computer Software and Applications Conference, 1999.
[4] Bruno, G., Agarwal, R. Torchiano, M. Static, dynamic and run-time modeling of compound classes, ACM SIGPLAN Notices, 31, 11, November 1996, 49-55.
[5] Infosys technical reference repository.
[6] Bruno, G. Agarwal, R. Modeling the Enterprise Engineering Environment, IEEE Transactions on Engineering Management, 44, 1, February 1997, 20-30.
[7] Catherine C. Marshall and Frank M. Shipman III, “Spatial Hypertext: Designing for Change,” Communication of The ACM, Vol. 38, No. 8, August 1996, pp. 88-97.
[8] Daniel E. O’Leary, “The Internet, Intranets, and the AI Renaissance”, IEEE Computer, January 1997, pp. 71-78.
[9] Hanafy Meleis and Racal Data Group, “Toward the Information Network”, IEEE Software, October 1996, pp. 59-67.
[10] Lance N. Ulanoff, et al., “Build Your Own WebSite, ”PC MAGAZINE, September 10, 1996, pp. 101-111.
[11] Michael Bieber and Fabio Vitali, “Toward Support for Hypermedia on the World Wide Web”, IEEE Computer, January 1997, pp. 62-70.
[12] Randall J. Atkinson, “Toward a More Secure Internet”, IEEE Computer, January 1997, pp. 57-61.
[13] Steven J. Vaughan-Nichols, “Switching to a Faster Internet”, IEEE Computer, January 1997, pp. 31-32.
[14] Tim Berners-Lee, “WWW: Past, Present, and Future,”IEEE Software, October 1996, pp. 69-77.
[15] Charles W. Krueger, “Software Reuse”, ACM Computer Surveys, page 131-183, June 1992.
[16] Debra J. Richardson, “TAOS: Testing with Analysis and Oracle Support”, International Symposium on Software Testing and Analysis, page 138-153, March 1994.
[17] R. Agarwal, G. Bruno and M. Torchiano, Domain Specific Software Architecture for Modeling and Simulating, presented at the Workshop on Object-Oriented Software Architecture, European Conference on Object-Oriented Programming (ECOOP) Brussels, Belgium, 20-24 July 1998.
[18] R. Agarwal, G. Bruno and M. Torchiano, Object-Oriented Architectural Support for Developing Complex Systems, presented at the IEEE Computer Software and Applications Conference, Phoenix, Arizona, Oct 27-29 1999.
[19] R. Agarwal, B. Ghosh, and A. Sarangi, Designing reliable software using DAF," in Proc. of FastAbstracts and Industrial Practices, 10th IEEE International Symposium on Software Reliability Engineering, Boca Raton, Florida, November 2-5 1999.
[20] R. Agarwal, G. Bruno and M. Torchiano, An Operational Approach to the Design of Workflow Systems in the Journal of Information and Software Technology published by Elsevier Science, Volume 42, Issue 8, pages 547-555, May 2000.
[21] The WEBLOAD user's guide, RadView Software.

Challenges and Experiences in establishing WebSite Quality

Rakesh Agarwal, Bhaskar Ghosh, Santanu Banerjee and Soumyendu Kishore Pal
Infosys Technologies Limited

Web Applications

- Extensive user interaction
- Extensive use of session state
- Layered architecture with diff. set of contents at different layers for ex.
  - User Interface layer
  - Business Logic Layer
  - Database layer

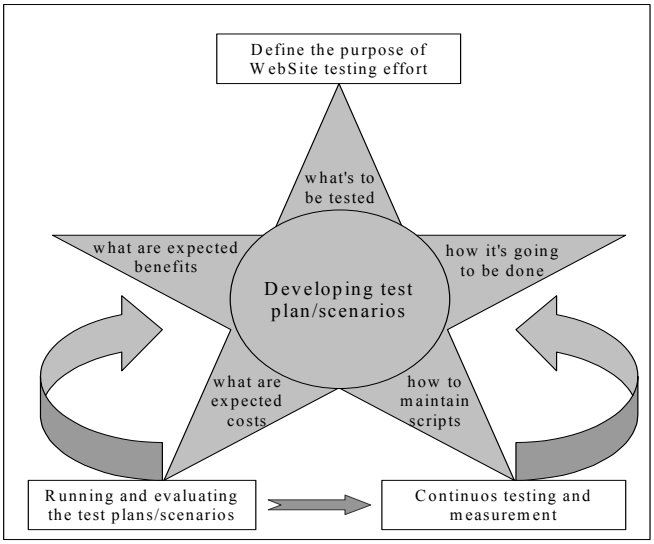
Testing Web Applications

- GUI intensive
- Database intensive
- Page execution order affects session state
- Rapid and continual change
  - Testing methodology must keep up
  - There may be no standard releases

Users in an web application

- Demography of users in a traditional app are generally known in advance. Their login pattern, usage pattern, general behavior, etc are known in advance. Generally, users on an intranet are reasonably predictable.
- Profile of users in an app exposed to the Internet is not known.
  - They could be genuine buyers, hackers, frivolous people, frauds, or people all out to have some fun.
  - Transaction rate is not known upfront; could vary by time of the day (or night!); there could be sudden peaks and troughs

Phases in testing WebSite



Quality Week 2000 Infosys Technologies Ltd 5

Process for testing the WebSite

- Define the purpose of WebSite testing effort.
- Develop test plan/scenarios.
- Running and evaluating the test plans/scenarios.
- Continuos testing and measurement.

Quality Week 2000 Infosys Technologies Ltd 6

Define the purpose of WebSite testing

- Time
  - WebSites change often and rapidly.
- Structural
  - How well do all of the parts of the WebSite hold together ?
- Content
  - Does the content of critical pages match what is supposed to be there?
- Accuracy and Consistency
  - Are today's copies of the pages downloaded the same as yesterday's?

Develop test plan/scenarios.

- Response Time and Latency
  - Does the WebSite server respond to a browser request within certain pre-planned parameters?
- Performance
  - Is the Browser-Web-WebSite-Web-Browser connection quick enough?
- Impact of Quality
  - Quality is in the mind of the user

Running and evaluating the test plans/scenarios

- The entire process of testing should be treated as a software development effort.
- This includes defining what should be automated, (the requirements phase), designing test automation, writing the scripts, testing the scripts, etc.
- The scripts need to be maintained over the life of the product just as any program would require maintenance.

Continuos testing and measurement

- The effort of testing is an investment.
- Sufficient time and resources are needed in order to obtain the benefits.
- Definitely, some scripts can be created which will provide immediate payoff, but this is not much of benefit.
- The benefit comes from running these automated tests every subsequent release.
  - Therefore ensuring that the scripts can be easily maintained becomes very important.

Assuring Website quality

- Browser Independent
- No Buffering, Caching
- Fonts and Preferences
- Object Mode
- Tables and Forms
- Frames
- Test Context

Website Validation Processes: Operational Testing

- Page Quality
  - Is the entire page identical with a prior version?
- Table, Form Quality
  - Are all of the parts of a table or form present?
- Page Relationship
  - Are all of the links a page mentions the same as before?
- Performance, Response Times
  - Is the response time for a user action the same as it was (within a range)?

Website Validation Processes

- Test Suites
  - Unattended Testing
  - Background Testing
  - Distributed Testing
  - Performance Testing
  - Random Testing
  - Error Recovery

Website Validation Processes

- Content Validation
  - Structural
    - All of the links and anchors match with prior "baseline" data
  - Checkpoints, Exact Reproduction
    - One or more text elements -- or even all text elements -- in a page should be markable as "required to match".
  - Gross Statistics
    - Page statistics (e.g. line, word, byte-count, checksum, etc.).
  - Selected Images/Fragments
    - The tester should have the option to rubber band sections of an image and require that the selection image match later during a subsequent rendition of it.

Website Validation Processes

- Load Simulation
  - Single Browser
  - Multiple Independent Browsers
  - Multiple Coordinated Browsers

Web Testing Theory

- Use existing test tools
  - Don’t reinvent the wheel
- Use database technology
  - Store, analyze, report test results
- Create a web testing website
  - Manage the testing effort, make it visible
- Think first – Automate second

Web Application Tests

- Links
- Forms
- Spelling
- Scripts
- Usability
- Performance

Link Testing

- Internal link errors
- External link errors
- “Page Not Found” errors
- “Host Not Found” errors

Link Testing

- Use the Forms Testing Checklists
  - Function Testing
  - Usability Testing
- Log User-Form Mistakes

Spell Checking

- Development tool spell checkers miss:
  - Server script generated text
  - Text from databases
  - Meta description and keywords text
  - Client script generated text
- Must spell check client rendered text

Scripts Testing

- Check web application specific scripts
- "White-box" script testing
- Check script vulnerabilities
  - Complex functions
  - Browser type specific code -- Netscape , IE etc
  - Browser version specific code
  - Date manipulation code

Usability Evaluation

- "Test by looking around"
- Create and test "Use Scenarios"
- Check for "zero training" capability
- Monitor web application session paths

Performance Monitoring

- Web hosting infrastructure
  - Capacity modeling and planning
  - Load balancing
  - Monitor web application subsystems
- Test for slow web pages
- Conduct Load/Stress testing

Representative Web Testing Products, Services (1)

- Website Testing Suites
  - Rational Suite Test Studio (Rational.com)
  - eConfidence (Segue.com)
- Tests e-business functions on the website
- Available on NT and Windows
- Price range $5500

Representative Web Testing Products, Services(2)

- Load/Stress Testing Suites
  - Active Test (merc-int.com)
  - Webload 3.5.1 (radview..com)
  - Rational Suite Performance Studio (rational.com)
- Stress, performance testing; uses document object model for virtual client ; Capacity Testing
- Available on NT ,Windows, Solaris
- Price range $15K- $35K

Representative Web Testing Products, Services (3)

- Performance analysis and Monitoring Suites
  - Perspective (keynote.com)
  - Topaz(merc-init.com)
  - Response Center (responsenetworks.com)
- Web application performance management product
- Available on NT and Windows
- Price range $3750/5 transactions per month

How to use a Web Stress Testing Tool like WebLoad

- Webload 3.5.1 simulates real world loads by generating virtual loads
  - Users create Java script based test scripts that define the virtual clients
  - Webload executes these test scripts and represents the results graphically in real time
- Webload has a console which can run a test session at console one can --
  - Define hosts participating in load session
  - specify a program that load session executes
  - Schedule tests

How to use a Web Stress Testing Tool like WebLoad

- The load machines are hosts which run the Load Generator software simulating multiple virtual clients
  - One load machine can run multiple load generators
  - The application being tested is where the web application being tested resides . Does not require a Webload installation
- The probing client machines are running Probing Client software acting as single Virtual Client and run at the same time as Load machines to further measure the performance .
- Webload uses Test talk -- to communicate between hosts and console


Thanks

QWE2000 Session 5M

Mr. Oliver Niese, Tiziana Margaria, Markus Nagelmann, Bernhard Steffen, Georg Brune & Hans-Dieter Ide
(META Frame Technologies GmbH)

An Open Environment for Automated Integration Testing

Key Points

- Automated, Integrated, Distributed Testing
- Testing Environments
- Test Management

Presentation Abstract

The increasing complexity of today's testing scenarios demands for an integrated, open, and flexible approach to test definition, evaluation, and management. Systems under test become integrated (e.g. include Computer Telephony Integrated platform aspects), embedded (e.g. with hardware/software codesign), and run on distributed architectures (e.g. client/server architectures). In addition, it is increasingly unrealistic to restrict the consideration to single units, since complex subsystems affect each other, and require scalable, integrated test methodologies.

In our approach, we add a Test Coordination layer driving the generation, execution, evaluation, and management of the system-level tests in the highly heterogeneous landscape. The coordination layer introduces the required flexibilization of the overall architecture of the test environment: it is a modular and open environment, so that diverse tools and units under test can be added at need. Through a CORBA/RMI-based implementation of the communication layer we are able to address and encapsulate a wide range of commercial test tools: this increases over time the reach and the capabilities of the resulting environment.

About the Speaker

Oliver Niese

Born 31.8.70, received the Diplom in Computer Science from the University of Dortmund (Germany) in February 1999. He is currently a Ph.D. candidate at the University of Dortmund and is working at METAFrame Technologies GmbH as a

consultant. He has experiences in large OO-projects with UML and Java as well as in formal verification methods. Within the ITE project, he is the technical leader at METAFrame.

Tiziana Margaria

Born 14.10.64, she holds a Laurea in Ingegneria Elettronica from the Politecnico di Torino (Italy) (1988) and a Ph.D. in Computer and Systems Engineering from the same institution (1993). She held assistant positions in Computer Science at the Technical University of Aachen and at the University of Passau. 1999 she was Visiting Professor at the Department of Computer Systems in Uppsala (S). Since June 1998 she is Senior Researcher at the University of Dortmund (D). She co-founded METAFrame Technologies GmbH in 1997, and serves since then as the company's CEO. Within the ITE project, Dr.Margaria in particular consults for formal verification methods.

Markus Nagelmann

Born 24.6.68, received the Diplom in Computer Science from the University of Dortmund (Germany) in April 2000. He is working at METAFrame Technologies GmbH as a consultant and gained experiences in computer telephony integration (CORBA, web based management) in projects with the business telephone switching division of Siemens ICN (Witten, Germany). Within the ITE project he is a member of the technical engineering team of the ITE project.

Bernhard Steffen

Born 31.5.58, he holds a Diplom in Mathematics (1983) and a Ph.D. in Computer Science (1987) from the University of Kiel (Germany). After holding research positions at the University of Edinburgh, UK (1988-89) and at the University of Aarhus, DK (1990) he was tenured Associate Professor at the Technical University of Aachen, D (1990-93) and held the Chair of Programming Systems at the University of Passau, D (1993-97). Since August 1997 he holds the Chair of Programming Systems at the University of Dortmund, D, where he heads the METAFrame project, concerning the development of a flexible environment for reliable definition and custom configuration of complex workflows. He co-founded METAFrame Technologies GmbH in 1997.

Georg Brune

Born 7.7.53, he holds a Diplom (1980) and a Ph.D. in Electrical Engineering (1985) from the University of Dortmund (Germany). After years of activity in hardware development (1985 û 1998), Dr.Brune heads since 1998 the CTI application development group within Siemens' Information and Communication Networks division (ICN EN HO SE 5). Dr.Brune heads the ITE project.

Hans-Dieter Ide

Born 18.7.52, he holds a Diplom (1979) and a Ph.D. in Electrical Engineering (1985) from the University of Dortmund (Germany). Since 1986 he works for Siemens AG.

Currently he leads the coordination and process development group within Siemens' Information and Communication Networks division (ICN EN HO SE 1).

An Open Environment for Automated Integrated Testing

O. Niese[1], T. Margaria[1], M. Nagelmann[1],
B. Steffen[2], G. Brune[3] & H.-D. Ide[3]

[1] METAFrame Technologies GmbH, Dortmund, Germany
[2] University of Dortmund, Dortmund, Germany
[3] Siemens AG, Witten, Germany

METAFrame Technologies GmbH

- Personalized Online Services
  - Online-Shops
  - Editorial Service
  - ...
- Telecommunication
  - Intelligent network service definition
  - **Testing of integrated system solutions**

Roadmap

- Introduction
- Integrated Testing
- Testmanagement
- Test formalism
- Realisation
- Conclusion/Further Work

Introduction

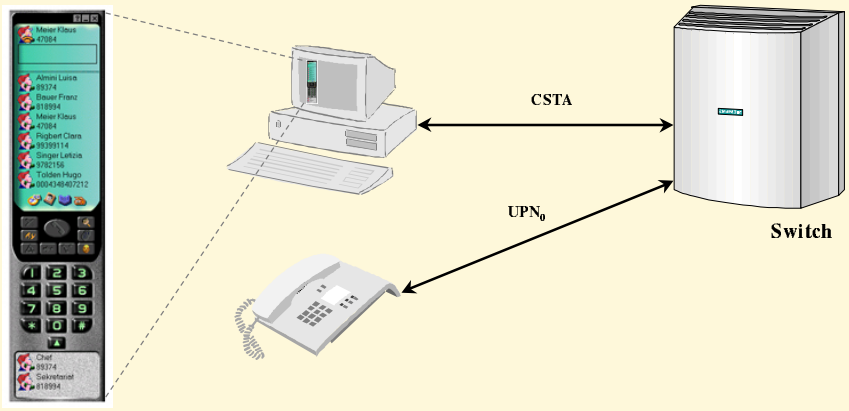
System-under-test become ...

- integrated
- embedded

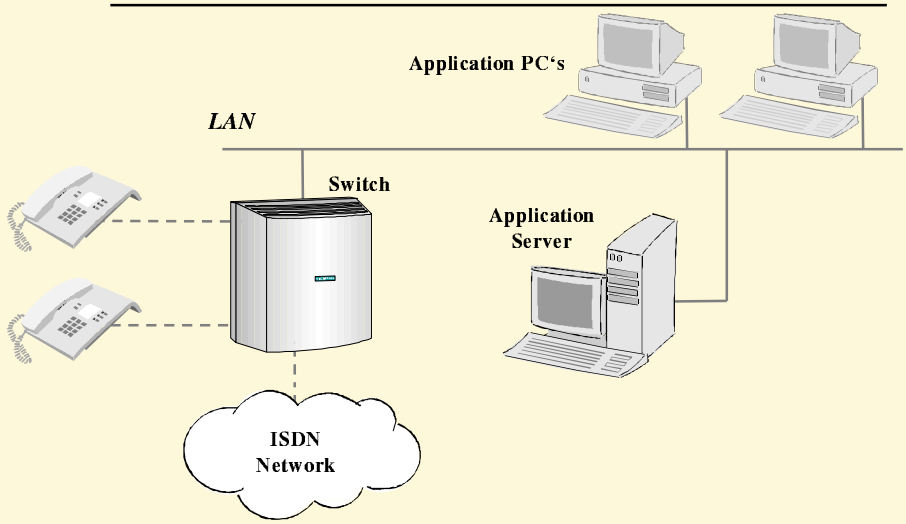
System-under-test run on ...

- distributed,
- heterogenous platforms

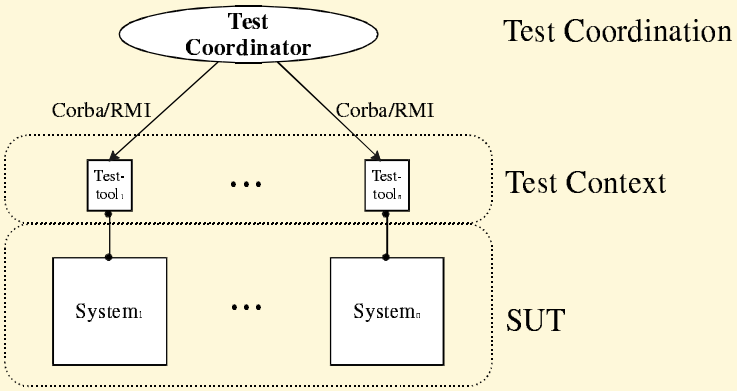
Typical Scenario



General Scenario



Integrated Testing



Testmanagement

- Organization
- Design
- **Coordination**
- Analysis

Organization

- **Version control** of all test relevant data
  - e.g. tests, configuration-files, documentation
- Configuration management
- **Structuring** of tests
  - e.g. avoid redundancies, compact representation

Design

- High-Level design of test cases
  - Intuitive modelling of test cases
  - Only application knowledge required
- Analysis
  - consistency checks of tests at design time
- **Automatic** generation of test cases
  - Generation of new test cases
- Variation of existing test cases

Coordination

- Coordination of heterogenous test tools
  - Initialization
  - Execution
  - Analysis
  - Documentation

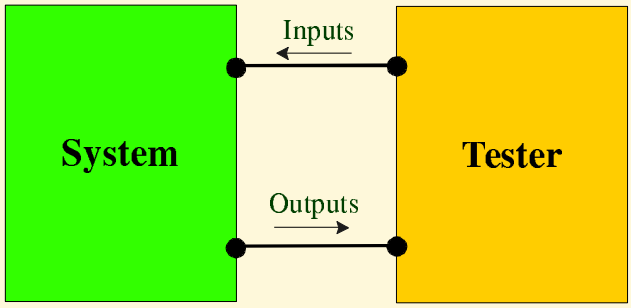
Analysis

- Error detection and diagnosis
- Analyse on abstract level → hide unnecessary details
- Detailled inspection must be possible

Testformalism

Adequate formalism
for the description
of these tests?

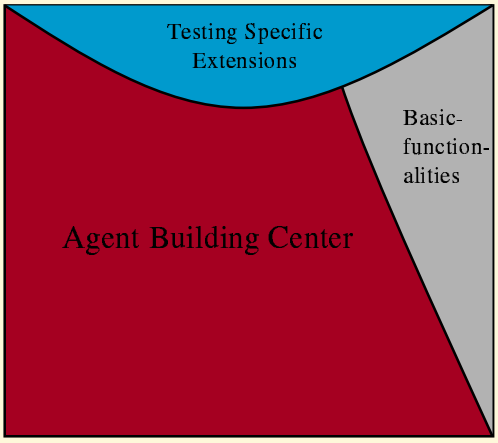
Black-Box Testing



Testformalism

- Provide basic functionalities for:
  - generic input stimulies (e.g. "press button")
  - output checkpoints (e.g. "is button enabled?")
- Combine basic functionalities to tests
  - graphical notation
  - describe control flow

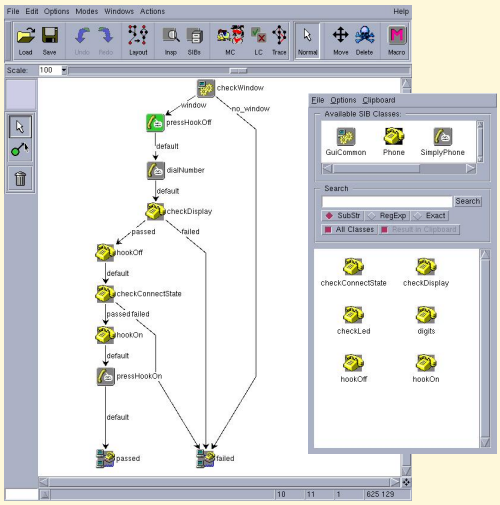
Test Coordinator



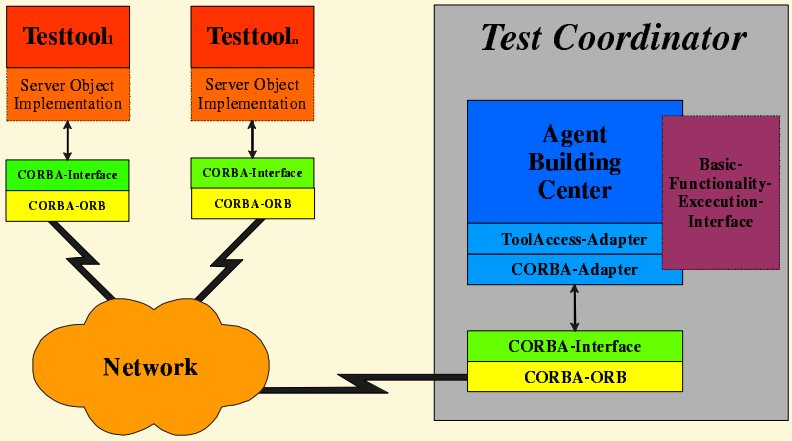
Agent Building Center

- Generic Workflow Management System
- Graphical Editor
- Analysis Mechanisms
  - libraries of constraints
  - automatic verification
  - automatic error location

Agent Building Center



Integration of test tools



Implementation of `ToolAccess`-Interface

- Direct implementation
  - e.g. vendor
- Plugin approach
  - e.g. plugins or libraries
- Separated server process
  - test tools offers remote access
  - external server process implements `ToolAccess`-Interface

Conclusion - Resumé

- Concept for integrated system-level testing
- Prototype with Siemens validates general concept
- Prototype shows practical impact
- Corba is suitable for coordination

Conclusion - Outlook

- Organization
  - Configuration management
  - Eliminate redundant tests
- Generation
  - Automatic combination/variation
- Analysis
  - Error detection/analysis

An Open Environment for Automated Integrated Testing

Oliver Niese [1], Tiziana Margaria [1], Markus Nagelmann [1], Bernhard Ste_en [2], Georg Brune [3] and Hans-Dieter Ide [3]

[1] METAFrame Technologies GmbH, Dortmund, Germany
{ONiese, TMargaria, MNagelmann} @METAFrame.de

[2] Chair of Programming Systems, University of Dortmund, Germany
Steffen@cs.uni-dortmund.de

[3] Siemens AG, Witten

{Georg.Brune, Hans-Dieter.Ide}@wit.siemens.de

Keywords:
Automated, Integrated, Distributed Testing; Testing Environments; Test Manage-Ment

Abstract:
The increasing complexity of today's testing scenarios demands for an integrated, open and exible approach to support the management of the overall test process. Furthermore systems under test become composite (e.g. including Computer Telephony Integrated platform aspects), embedded (e.g. with hardware/software codesign) and run on distributed architectures (e.g. client/server architectures). In addition, it is increasingly unrealistic to restrict the consideration of the testing activities to single units, since complex subsystems a_ect each other and require scalable, integrated test methodologies. In this paper, we present a test management layer driving the generation, execution and evaluation of system-level tests in a highly heterogeneous landscape. The management layer introduces the required exibilization of the overall architecture of the test environment: it is a modular and open environment, so that several tools and units under test can be added at need. By means of a CORBA/RMI-based implementation of the external interfaces of the management layer, we are able to address and encapsulate a wide range of commercial test tools. This increases over time the reach and the capabilities of the resulting environment.

1 Introduction

The increasing complexity of today's testing scenarios demands for an integrated, open and flexible approach to support the management of the overall test process, e.g. specification of tests, execution of tests and analysis of test results. Furthermore, systems under test (SUT) become composite (e.g. including *Computer Telephony Integrated* (CTI) platform aspects), embedded (e.g. with hardware/software codesign) and run on distributed architectures (e.g. client/server architectures). In addition, it is increasingly unrealistic to restrict the consideration of the testing activities to single units of the systems, since complex subsystems affect each other and require scalable, integrated test methodologies.

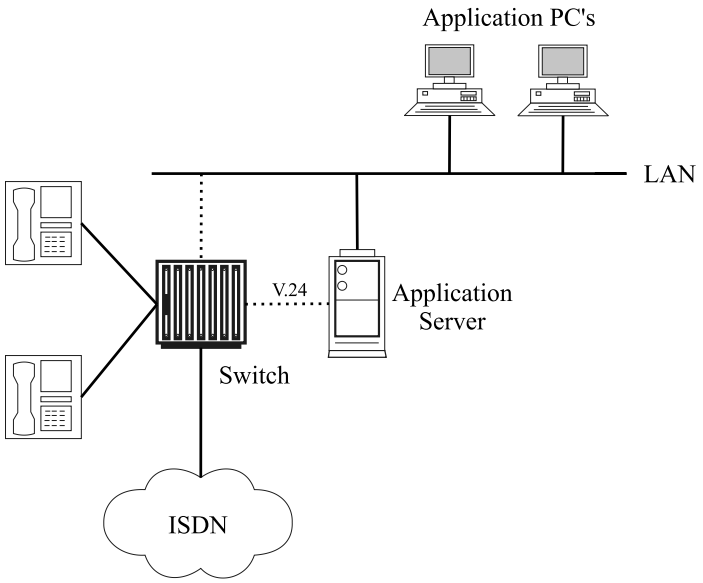


Fig. 1. Example of an integrated CTI platform

As an example for an integrated CTI platform, fig. 1 shows a telephone switch and its environment. The switch is connected to the ISDN telephone network, or more generally to the public telephone network (PSTN) and acts as a "normal" telephone switch to the phones. Additionally, it communicates directly via a LAN or indirectly via an application server with CTI applications that are executed on PCs. The CTI applications are also active components, like the phones. So it is possible for an application to control the switch (e.g. initiate a call) and vice versa (e.g. notification of an incoming call). In a system test, it is important to investigate the interaction between the subsystems.

More generally speaking, in our context a typical system under test is composed of several independent subsystems communicating with each other. To test this kind of composite systems we need to resort to specific test tools for each participating subsystem. Thus, we must be able to coordinate heterogenous test tools in a context of heterogenous platforms, a task exceeding the capabilities of today's commercial test management tools.

2 Overview

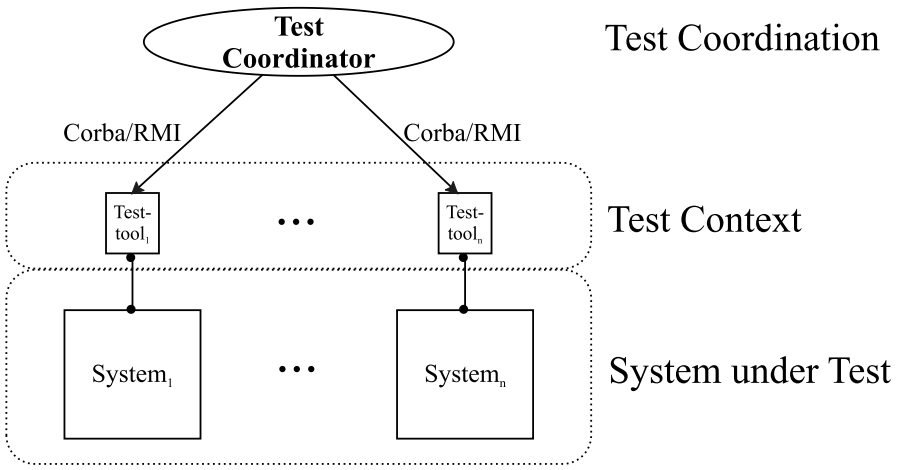


Fig. 2. Architectural Overview of the Test Environment

Figure 2 shows the general architecture of the test environment. The system under test (SUT) is composed out of several subsystems communicating with and affecting each other. The subsystems can be hardware or software components, often also including and driving external applications or devices. Each of the components can also be used in completely different configurations in other scenarios. In general each subsystem has to be tested using its specific test tool or at least using its own instance of a test tool. To coordinate the different control and inspection activities of an integrated test a test management layer is mandatory. This test management layer, called *Test Coordinator* in fig. 2, communicates with the test tools by means of **C**ommon **O**bject **R**equest **B**roker **A**rchitecture (CORBA) [3] or **R**emote **M**ethod **I**nvocation (RMI) [4].

However, when testing composite systems it is not sufficient to support the aspect of coordinating test tools only, but the whole process, from test specification to the analysis of test results. Therefore, the following aspects of the test process have to be supported by an integrated test environment:

1. *Organization* of test relevant data
2. *Design* of test cases and composition of test suites
3. *Coordination* of test execution
4. *Analysis* of test execution results

2.1 Organization

The organizational aspects of the test process are among others:

Version control Beside the test cases itself many other files have to be organized throughout the test process, e.g. configuration files or test documentation. Because of changes throughout the test process it is important to capture the history of changes and dependencies between versions.

Configuration management It is mandatory, especially when considering integrated tests, that the system under test is in a well defined state before executing tests. This is a non-trivial task because we treat complex systems, where the initialization of one component can affect the state of other components. Moreover, it is also important to document the versions of the subsystems and test tools and to ensure that they are correctly working together.

Structuring of tests Tests have to be structured to
- provide a simple mechanism to build test suites out of the set of test cases via criteria, e.g. *regression test* or *feature test*
- eliminate redundant test cases which may dramatically reduce the whole test execution time.

2.2 Design

The design of test cases, i.e. specifying which control or inspection activities have to be performed and in which order, should be possible without any expertise of how to apply a specific test tool. Therefore, the design should be intuitive but reliable concerning executability and other frame conditions.

Design Test cases are specified on the level of SUT-usage and are formulated in a graphical way. Hierarchical design, i.e. the usage of a macro mechanism should be possible.

Analysis Consistency checks of tests cases at design time to ensure the correct specification of the test cases.

Automatic generation Beside the manual design of test cases, automatic derivation of new test cases out of the system specification should be possible.

Variation The automatic variation of existing test cases should be possible by means of parameter variation.

2.3 Coordination

The whole test execution process must be supported, including:

Initialization of the whole test scenario (SUT components, test tools)
Execution of the test cases, i.e. instructing the test tools
Analysis of the results of test runs
Documentation of the test runs and their results

2.4 Analysis

The analysis of the results of test runs for the error detection and diagnosis is an essential feature of every test environment. It must be possible to analyse results on an abstract level, in order to hide unnecessary details. However, if required an inspection of the details must be possible.

3 Test formalism

When providing such a wide support for test management an adequate formalism for the description of tests is mandatory. Since we are considering system-level tests, we are in fact performing black-box testing. Thus we are abstracting from the internals of the system and take only external observable "inputs" and "outputs" of the system into account, cf. fig. 3.

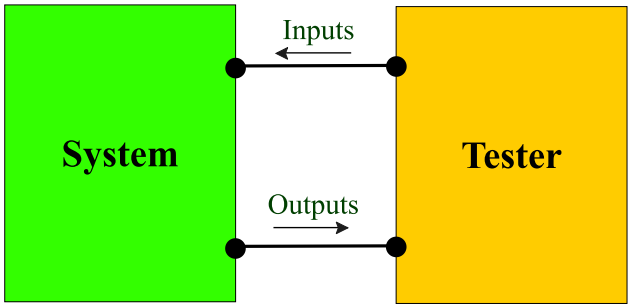


Fig. 3. Black-box testing

In order to specify test cases at this abstract level, i.e. to allow a flexible test modelling, we build test cases out of generic basic functionalities. Such a basic functionality can be either a single *stimulus* for the system (input) or a single *check point* which checks the status of the system (output). In the context of testing an application these basic functionalities may be e.g. *"pressing a specific button on the GUI"* or the check *"is a specific button enabled?"*.

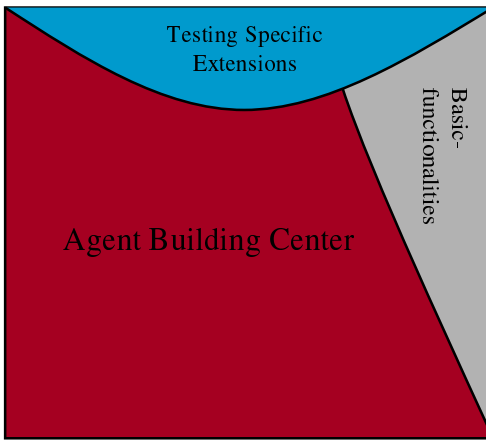


Fig. 4. Test coordinator as instance of the *Agent Building Center*

4 Realisation

The heart of the test management environment is the *Test Coordinator*, built on top of METAFrame's *Agent Building Center* (ABC) [1,2], which is a generic and flexible workflow management system. In this application, we view system-level test cases as executable workflows within the integrated test environment (which plays the role of an extended runtime environment). Building on the ABC's capabilities, test cases can be graphically designed from palettes of subsystem-specific, generic test components or basic functionalities. Test cases are combinations of these basic functionalities, which are connected through edges to describe the control flow, see fig. 5.

The main portion of the test coordinator is the ABC itself, cf. fig. 4. Basically, it offers the functionalities necessary to cover the organizational and the design related aspects. Some extensions are needed before the ABC can be used as a test coordinator:

Testing specific extensions Among others communication with specific test tools must be integrated into the ABC.

Basic functionalities Stimuli and check points which are necessary to test a SUT's functionality must be provided as basic functionalities. They will be implemented using the specific commands provided by the relative test tool.

While the basic functionalities are developed by experts of the particular system, the test cases can be designed graphically at an intuitive level by testers.

The test coordinator uses the capabilities of the ABC e.g. for the design and analysis of the test cases.

Figure 5 shows how test cases can be designed within the ABC. On the left hand side is the editor, which allows the design of test cases. On the other side is a browser for the available basic functionalities which are structured into classes[1]. In this example there are three classes of basic functionalities: *GuiCommon*, *Phone* and *SimplyPhone*.

The class *GuiCommon* provides generic functionalities for the usage of a Graphical User Interface, e.g. `checkWindow` checks whether a specific window is visible on the desktop or not. The class *Phone* provides input-actions (e.g. `hookOff`) and output-actions (e.g. `checkConnectState`) which allows the tester to control a "real" phone, whereas the class *SimplyPhone* can be used to test a specific application called SimplyPhone which is a CTI-Application that simulates a phone.

The analysis methods of the ABC are based on formal verification methods (model checking). In particular,

- *libraries of constraints* capture the essence of the test designers' expertise about do's and dont's of test definition. Automatically accessed by the model checker during the verification, they express vital properties concerning the interplay between the components of a test as global correctness and consistency conditions of the test logic.
- *automatic verification* of test-dependent frame conditions (from the constraint library) allows designers to verify that the test is consistent **before** executing it,
- *automatic error location* delivers the exact portions of the test where violations of frame conditions occur. This eliminates the manual search for the erroneous segments in the test. This can be used either at design time (see above) or at runtime, when analysing erroneous test runs.

5 Integration of test tools

To communicate with different test tools the test environment offers a general CORBA interface: `ToolAccess`, cf. fig. 6. This interface comprises basic methods (e.g. `getName`, `getVersion`) which all test tools have to support. Special features (e.g. input and output commands for testing a special SUT component) can be added by extending the `ToolAccess` interface. From the basic functionality execution environment the special methods of the test tool are accessible for the tester via a test tool specific adapter.

The extension of the ABC environment through the integration of adapters is the key to our approach.

[1] Note that each class is represented by its own icon, so that they can be distinguished easily in the graphical notation.

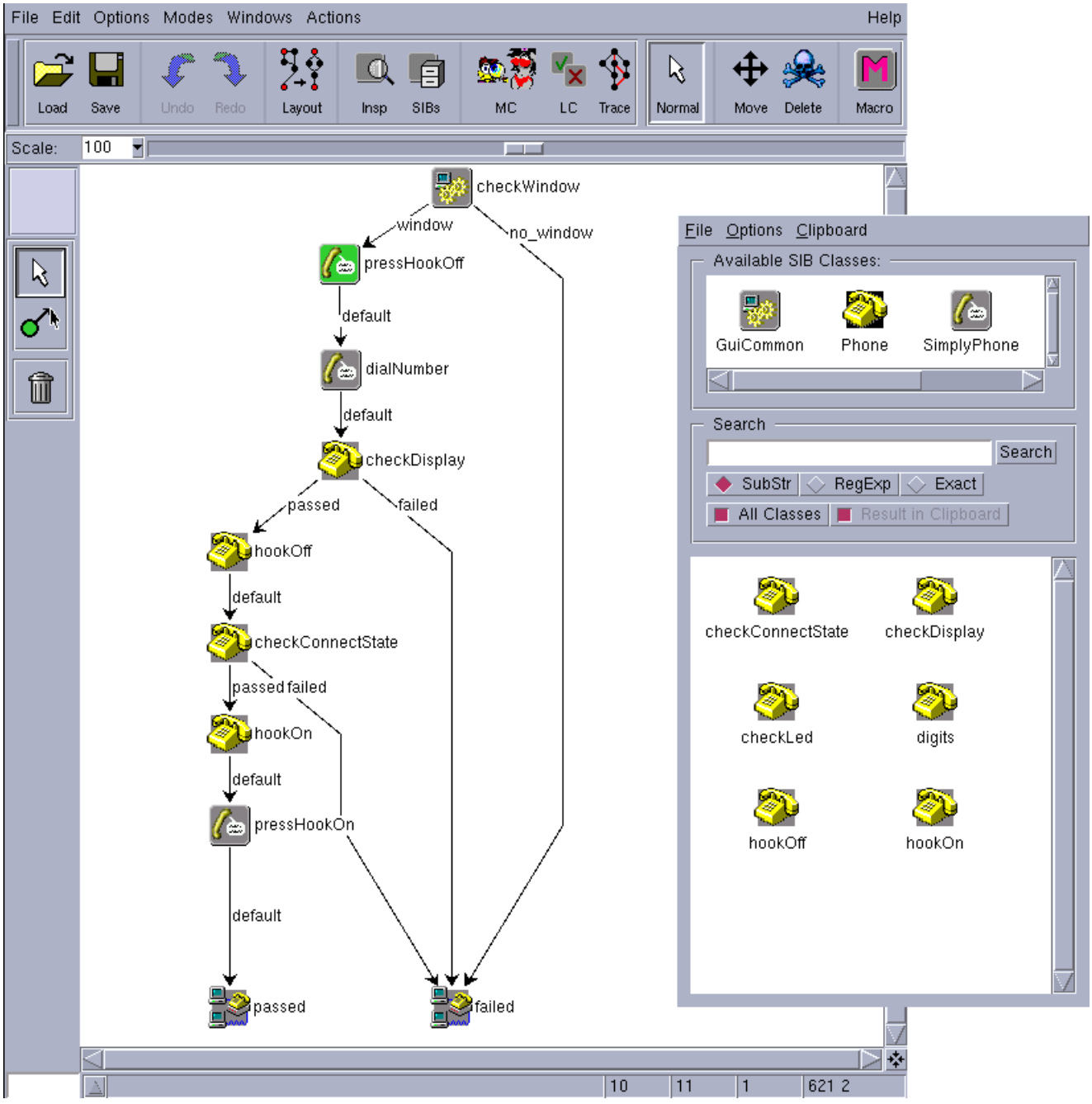


Fig. 5. Example of a test case

The integration process consists of two main activities:

1. Integration of the interface provided by the test tool into the test coordinator and
2. Implementation of the interface functionality by the test tool.

When the `ToolAccess` interface is not already implemented in the test tool by the vendor, there are different ways to implement it:

Plug-in approach If the test tool supports customisation via loading plugins or libraries, the `ToolAccess` interface can be integrated by implementing such a plugin including the CORBA object request broker.

Separated server process If a test tool offers remote access via an interface of its own (e.g. COM, DCOM or CORBA) a separate server process can communicate via this interface with the test tool, cf. fig. 6. Then, the special server implements the interface or its derivation and communicates with the test coordinator, i.e. it can be seen as a relay between the test coordinator and the test tool.

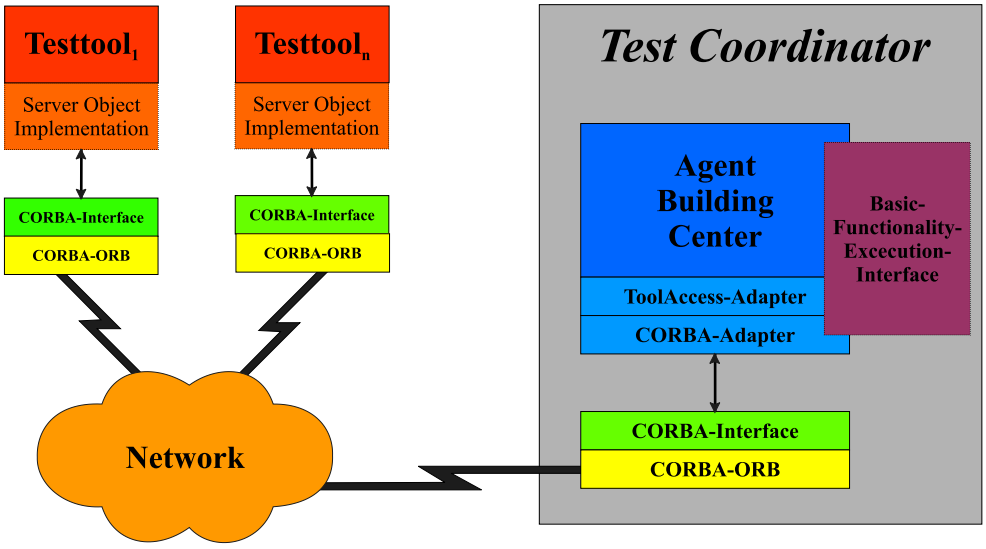


Fig. 6. Integration of test tools

6 Conclusion

To test composite systems in an integrated way, we have added a *test management* layer responsible for the generation, execution and evaluation of system-level tests

in a highly heterogeneous landscape. The management layer introduces the required flexibilization of the overall architecture of the test environment: it is a modular and open environment, so that several tools and units under test can be added at need. By means of a CORBA/RMI-based implementation of the communication layer we are able to address and encapsulate a wide range of commercial test tools: this increases over time the reach and the capabilities of the resulting environment.

Beside the test tool coordination described in this paper, we will support all other important aspects of the test process in a more comprehensive way in the near future:

1. *Organization* of test relevant data
2. *Design* of test cases and building of test suites
3. *Analysis* of test execution results

Together with Siemens ICN an integrated test environment has been developed to test the interaction between a medium-range telephone switching system and a variety of computer-telephony applications to prove the concept.

Acknowledgements

We would like to thank the whole *ITE-Team*, especially Andreas Hagerer for helpful comments and criticism on draft copies.

References

1. B. Steffen, T. Margaria: *METAFrame in Practice: Intelligent Network Service Design*, In *Correct System Design – Issues, Methods and Perspectives*, E.-R. Olderog and B. Steffen (eds.), LNCS 1710, Springer Verlag, 1999, pp.390-415.
2. B. Steffen, T. Margaria: *Coarse-grain Component Based Software Development: The METAFrame Approach*, invited to the 3. Fachkongress "Smalltalk und Java in Industrie und Ausbildung" (STJA'97), 10.-11. September 1997, Erfurt.
3. *The Common Object Request Broker: Architecture and Specification*, Revision 2.3, Object Management Group, 1999.
4. *Java(TM) Remote Method Invocation*, Sun, *http://java.sun.com/products/jdk/rmi*.

QWE2000 Vendor Technical Presentation VT5

Mr. Alexander Malshakov & Mr. George St. Clare
(Amphora Quality Technologies)

Optimizing Iteration Testing

Key Points

- How to use efficiently modern object oriented SQA techniques
- Optimum using of Test Outsourcing
- Flexible management under challenging testing

Presentation Abstract

Since the dawn of the Internet and e-business era software developers have been facing the problem of time pressure and human resources shortage. Traditionally a software product testing is conducted just before delivering to the customer so Testing departments of software development companies suffer the greatest stress. After all it considerably downgrades software quality. The employment of modern object-oriented SQA methodologies introducing software-testing phase at an earlier development stages improves software quality. In contrast to the standard “waterfall” process this class of methodologies is based on iterative approach to testing process with gradual widening of Test Cases range. However, large number of iterations creates counter problems: the necessity of additional personnel training, lack of automation and inflexibility of management. SQA Outsourcing is a good solution to this case that allows a software development company to make use of a highly professional SQA team and advanced automation tools. But test outsourcing procedure should meet the internal development cycle requirements and test iteration optimization is the key to outsourcing efficiency.

About the Speaker

Alexander Malshakov is co-founder and Managing director of Amphora Quality Technologies, the very first Russian private company that has become leading Russian SQA services provider since last year. Within 7 years he made career from a system programmer to General Development Manager of a bank automation system. Working for Diasoft Ltd. he was responsible for development of the most popular Russian bank automation system, that has had more then 600 installations by now. He has a number of publications devoted to software development process. Perceptions of necessity of profound approach to testing and quality assurance led

him to the foundation of AQT Company. Alexander's high weight among Russian software development community allowed him to team up highly professional and skilful specialists and managers. AQT is recognized as a reliable partner not only in Russia but also in the UK and the USA.

AQT Services

Introduction

Amphora Quality Technologies believes that quality has strategic meaning for any business. With AQT as your partner in Software Validation and Verification, you can improve product quality, free up technical and intellectual resources, lower overall development costs and develop your products faster than ever before.

AQT is able to offer a well-established service which encompasses the entire project lifecycle and which allows the delivery of a competitive product to customers. Specialized company divisions – Web, Functionality and Performance Labs as well as a Research Department - provide comprehensive Quality Assurance and Testing services to Software Developers and Corporations worldwide.

Application domains

Amphora Quality Technologies has particular expertise in QA and Testing of the applications of the following types:

- **Intranet/Extranet Information Systems;**
- **e-Business solutions;**
- Corporate Web Sites;
- **Banking and Finance solutions;**
- Client-Server business applications;
- **Electronic document management and Workflow;**
- Data Warehousing solutions;
- **Application servers & Middleware's;**
- Networking solutions;
- Geo Information System (GIS) - based solutions;
- Multimedia applications;
- Object Oriented development tools

Consulting and Quality Assurance

The provision of software quality assurance is a complex process, requiring highly qualified personnel, a well-run development system and an understanding of prioritization. However, this is all just part and parcel of success in this field. It is enough to competently utilize the rules of the selected development methodologies and control quality.

Amphora Quality Technologies provides comprehensive assistance and consultation in respect of organization of processes resulting in high-quality software. AQT analysts are very well-versed in the object-oriented engineering methodologies of the Rational Unified Process software at the topmost professional level. They are experienced in providing quality in the biggest software projects. Our research department personnel carry out their own scientific studies directed towards development of methods of analysis and forecasting according to research test results.

The development of high quality systems requires a series of special measures to be initiated right at the beginning of the development cycle. AQT's analysts will either develop a quality assurance strategy for you, or help you create this strategy independently. We would be happy to implement proven software analysis, design, development and testing methodologies for your organization.

AQT analysts will perform an appraisal of project documentation and system architecture. Our experienced consultants will propose recommendations and specialized program tools for the

organization of project database, management configuration and the performance of automated testing. AQT will assist in the effective organization of information exchange between your team members and will oversee correspondence between UML model documentation and RUP requirements. Company experts will develop and train your personnel in the use of metrics. This will result in the best methods for determining your software characteristics and forecasting their change dynamics.

If necessary, we are prepared to take on test investigations and organize regression testing, along with test results interpretation and analysis of problems detected. In close cooperation with your specialists, AQT's experienced consultants will develop an individual software testing program. By taking into account technical project characteristics and priority requirements in respect of the software, our testing experts will help you to determine exactly which tests and investigations must be performed for effective localization and elimination of problems.

Software Testing

Amphora Quality Technologies performs a full spectrum of tests and investigations directed towards the measurement and evaluation of various software quality characteristics. It does not matter if your system is ready for delivery to the client or you have just started work on a prototype. We will provide exhaustive and reliable reports regarding project requirements, performance and reliability, along with many other quality parameters. In addition to testing, AQT performs architectural and technical solution analysis, plus project documentation.

Contact us and AQT analysts will develop an integrated testing and quality assurance strategy for your IT project. They will also help you evaluate and refine measures you have already taken.

Tests & investigations

Software quality evaluation by AQT specialists means analysis of project materials, requirements and priorities, planning and performance of test investigations and appraisal of the results obtained. All project stages are accompanied by appropriate **Rational Unified Process** and **Unified Modeling Language model documentation**, systematizing the information into a convenient format.

Upon test conclusion, investigation results are summarized and presented as a formal report. This report contains a detailed description of work performed and a list of problems detected. For easy comprehension, unrefined test results are presented with graphics and diagrams, which clearly demonstrate the characteristics obtained at a glance. Apart from this, the report contains an expert evaluation of software quality prepared by AQT analysts. If necessary, comparative analysis of system characteristics is performed with previous versions or selected standards.

Test process organization and quality assurance by our company is described in detail in the QA & Testing process section.

Below is a list of tests and investigations performed by AQT's Web, Functionality and Performance Laboratories.

Web Site and Internet Application testing

Building industrial e-Business solutions involves a greater variety of technologies and tools than other software application development projects. Naturally, analysis and software quality assurance for the Internet require the application of new methodologies and means of automation.

AQT's specialized division – the Web Laboratory – has all the essential resources for performance of integrated quality analysis of a broad spectrum of on-line solutions. Tried and tested technology for the performance of test investigations provides for reliable results, which, in conjunction with highly-qualified lab experts, allows comprehensive cause analysis of problems detected. The use of Rational Unified Process methodologies and broad-band methods of test automation allow us to work quickly and effectively.

AQT offers a full range of Internet oriented QA & Testing services:

- **Functionality and Interface Testing** - Validation of the entire site/application, ensuring that there are no invalid content, functionality or application errors; Checking of visual design in accordance with corporate standards (if requested); Checks may be performed from release to release (Regression testing);
- **Website Integrity Testing** - Detection of broken links and orphaned pages, verification of accessibility of dynamically formed data;
- **Active Content Verification** – Identification and examination of HTML forms, Java applets, Scripts, plug-ins, Active-X controls and others active elements of a Web site;
- **Performance, Load and Stress Testing** – Determination of application performance characteristics by means of simulation of a large number of users carrying out real-life transactions in a Web application or surfing the Website; AQT's specialized **Performance Lab** conducts a wide range of investigations, allowing for the provision of exhaustive and reliable information on the performance parameters of the Internet solution;
- **Availability and Reliability Testing** - Continuous, comprehensive testing and monitoring of Web sites while they are in use, twenty four hours a day, seven days a week;
- **Compatibility testing** – Verification of Software operation across the range of hardware and software configuration (Operating systems, Browsers) for which it was designed; There are over 150 possible combinations of different widely used client-side operating systems and various versions of Netscape, Internet Explorer and other browsers. It is important to test across a large number of these to ensure that users with diverse configurations don't experience problems when using your Web site or application.
- **Database operations analysis** – Monitoring of database activity, checking all queries and transactions between a Web application and SQL Server; identifying where and why transactions may be performing poorly;
- **Server Side Testing** – Analysis and Monitoring of Functionality and Performance parameters of software components, functionality within the framework of a Web-server, server applications, database transaction monitoring. Examples of server side testing include checking database integrity on the database server itself, verifying that Active Server Page (ASP) scripts are being executed correctly on the server and determining how well a Web site functions when run on different types of Web server;
- **Integration Analysis** – Investigation of structure and filling of inter-component information streams in multi-level Internet applications with the goal of discovering bottle-necks and non-optimal solutions from the point of view of functionality, reliability and performance. Examples of Integration Analysis include testing the interaction between a Web and an application server, checking that the business-logic interfaces provide appropriate performance and assessment of the efficacy of database access methods;
- **Network Traffic Analysis** – Capturing and monitoring Network traffic, analysis of Communications Efficiency;
- **Security and Vulnerability Testing** - Active probing of the system Network and Server components to identify potential vulnerabilities, which could damage an on-line database or even the whole system. Checking of firewall configuration, set-up of network, servers, and databases.

WAP testing

The popularity of WAP services is growing rapidly.

AQT's Web Laboratory performs comprehensive testing of WAP-sites and applications for compatibility and usability with different WAP browser models (each piece of apparatus has its own browser, i.e. differing screen sizes and number of lines). We also perform other forms of testing for WAP-sites such as performance testing. Today, your WAP site may not have many visitors, but the number of WAP service users is growing exponentially and, tomorrow, your site may be unable to cope with the number of visitors wishing to use it.

Functionality analysis

Software quality is a complex integrated indicator. Qualitative evaluation of quality is achieved through the application of metrics connected with functionality, reliability, performance and many other characteristics. However, the most important quality criterion, naturally, is functionality correspondence between the system and project specifications. In other words, system adequacy with respect to given requirements.

Analysis of system functionality and evaluation of correspondence to requirements is performed by AQT's Functionality Laboratory specialists. Functionality analysis means performance of the following tests:

- **Functionality Testing** (black box testing) – Checks product operation against its functional specification to ensure that operation is as designed. This test can be quite simple to ensure primary functional operation, or as detailed as checking a variety of scenarios and validating that all output meets specified expectations;
- **Defect Analysis** – Identifies any area of software operation which may reduce product quality;
- **Visual Interface Testing** - Automated Testing of visual forms functionality and checking design is in accordance with corporate standards (if requested);
- **System Components Testing** – Functionality Testing of individual Component operation, including examination of DCOM, CORBA and Java RMI components as well as other components present as Source Code;
- **Compatibility and Portability Testing** – A software product is tested across the range of platforms, operating systems and hardware and software configurations for which it was designed to ensure the system functions as intended; In particular, AQT performs compatibility testing between applications in the **Windows 2000 Application Specification**;
- **Interoperability Testing** – Investigations directed towards the evaluation of system processes interaction with external components. To be more precise: specific hardware, device drivers and second-party software, etc. As a rule, we are talking about testing reliability and performance characteristics of inter-system gateways, and also evaluation of effectiveness of architectural and technical solutions.

Cause analysis of discovered defects may be performed according to the results, as might statistical analysis of the software's qualitative quality characteristics. The use of the qualitative metrics method of testing in conjunction with regression testing (release to release), allows us to determine the dynamics of software quality changes and makes the software development process manageable and foreseeable.

Performance & Load Analysis

Performance and throughput is one of the most important characteristics of modern information systems. For multi-national corporations engaged in the world of e-commerce and on-line solutions, where thousands of transactions are processed by the system in a few seconds, scalability and stability of peak loads become essential elements for success.

AQT's Performance Lab is a team of specialists highly experienced in software testing, analysis and performance optimization. The leading lab specialists already have more than ten years experience in precisely this sphere and are experts in performance optimization of large systems on diverse platforms – from PC to IBM AS/400.

The main goal of the lab team is the detection and localization of "bottle-necks" and non-optimal solutions in the system. AQT specialists analyze the architecture and technical solutions used by the system developers, perform test experiments and also create temporary profiles and perform component debugging.

AQT's Performance Lab has unique experience in Internet and multi-level Client-Server architecture performance optimization. Among our clients in this sphere are well-known software developers such as Informix Software. The broad spectrum of basic tests performed by the lab and also the series of specialized tests and measurements - Middleware server/component Testing, Communication and Integration Analysis, White Box Testing & Source Code Analysis and Cause Analysis, plus test results interpretation – allow us to reliably identify and appraise possible paths to performance optimization.

Software quality analysis from the performance standpoint includes the following tests:

- **Benchmark testing** - Tests that use a standard, reference workload to measure the performance of a system and compare it to a known reference system (or measurement);
- **Performance testing** - Tests using a constant workload and varying system variables and environment configuration to verify the acceptability of the target-of-test's performance behavior and tune (or optimize) it. Measurements typically include the number of transactions per minute, number of users, and size of the database being accessed;
- **Load testing** - Tests to verify and assess acceptability of the operational limits of a system under varying workloads while the system-under-test remains constant. Measurements include the characteristics of the workload and response time;
- **Distribution and Load balancing Analysis** - When systems incorporate distributed architectures or load balancing, special tests are performed to ensure the distribution and load balancing methods function appropriately;
- **Contention Test** - Verifies the system can acceptably handle multiple user demands on the same resource (data records, memory, etc.);
- **Volume Testing** - Testing that focuses on the ability of the system to handle large amounts of data, either as input and output or resident within the database;
- **Stress Testing** - Tests that focus on ensuring the system functions as intended when abnormal conditions are encountered. Stresses on the system may include extreme workloads, insufficient memory, unavailable services/hardware, or diminished shared resources;
- **Scalability Testing** - Tests to measure and analyze speed of product operation on different hardware/software platforms and database management systems.

Reliability testing

- **Integrity Testing** - Tests which focus on assessing the system's robustness (resistance to failure) and technical compliance to language, syntax, and resource usage;
- **Reliability Testing** - Tests product operation within a continuous working period under conditions of heavy loading and high volume.

Security analysis

- **Security Testing** – Checks database and communications security level and conducts overall analysis of architecture and quality of product's security sub-system;
- **Vulnerability Testing** – Tests focused on a search for potential Software Vulnerabilities leading to unauthorized access to information or system faults.

Middleware server/component Testing

Middleware constitutes one of the most important components of contemporary information systems and e-Business solutions. As a rule, business logic and specialized inter-component gateways function within the Middleware framework, providing an effective interface between program layers. The quality and performance of the server system at an intermediate level are largely determined by the corresponding software characteristics as a whole.

AQT has extensive experience with development of Middleware test suites, particularly Transaction monitors, Application servers and Component Object Request Brokers. One of our clients in this sphere is Informix Software.

Middleware testing includes Functionality, Performance, Interoperability, Security and other tests. Test subjects include the Middleware server itself, application business components and the communication subsystem. Middleware testing is performed by professional developers, AQT's Performance Lab's most experienced specialists. It involves the creation of specialized auxiliary software systems and components allowing full information about server performance characteristics in different regimes to be obtained, and also allows the construction of temporary component profiles.

This unique expertise makes AQT ideally qualified to design and implement tests for N-tier distributed applications and middleware.

Communication and Integration Analysis

AQT has successfully completed a series of projects in respect of analysis and optimization of performance of multi-level systems in client-server architecture. Quite often, "bottle-necks" are discovered in the communication subsystem or at the point where software layers interface. AQT's team uses special equipment to analyze interaction performance between business components of distributed systems.

Apart from load testing, AQT's Performance Lab performs the following tests:

- Network Traffic Analysis – investigations directed towards the evaluation of network exchange performance between distributed components of heterogeneous systems; this analysis is subject to the architecture and technical solutions used upon creation of the communications subsystem;
- Integration analysis – investigations of structure and filling of inter-component information streams of multi-level object-oriented systems with the goal of detecting bottle-necks and

non-optimal solutions; Analysis subjects include system interface components, and also the quality and constitution of messages/calls transferred.

The Functionality Lab performs additional experiments and analytical investigations in this sphere, allowing the provision of great accuracy and reliability of investigation test results. Furthermore, this allows cause analysis of problems detected.

Usability testing

- **Usability Testing** – Testing in a true end-user environment in order to check whether the system is able to operate properly in accordance with the exact set of processes and steps applied by the end-user, including user's interface and system convenience estimation;
- **Installation, Update and Configuration Facilities Testing** – Determines how well and how easily a product installs and updates on a variety of platforms, software configurations and under different conditions (such as insufficient disk space or power interrupt);

Technical requirements analysis

- **Compliance analysis** - Checking the accuracy and completeness of technical requirements set by developers to software operation environment;
- **Requirements correctness testing** – General Functionality, Performance and Reliability analysis of system operation on various hardware and software platforms and in various configurations regulated by technical requirements

White Box Testing & Source Code Analysis

The presence of source code and software technical specifications broadens the possibilities for a series of test and analytical investigations, and also significantly increases their results reliability. Essentially, this may be explained in that there is access to all information pertaining to the architecture and to the actual way in which components were created. In conjunction with this, there is a series of specialized investigations, whose execution, in principle, is not possible if the system is viewed as a black box.

AQT's Functionality Lab experts perform a series of tests and investigations using information available on the architecture and software source code. These are mainly directed towards attainment of greater accuracy and checking the reliability of functionality and load test results, and also cause analysis of problems detected during testing.

Some of the measures performed by AQT in this sphere are listed below:

- Evaluation of internal architecture and particularities of system creation – an investigation directed towards a search for non-optimal architectural, technical and program solutions used in project preparation and system development; Analysis presupposes a series of tests, and also the manual study of system specifics and source code;
- Computation and analysis of code based metrics of software quality and test coverage – correspondence of different system quality indicators and completeness of testing performed with individual components, procedures and functions;
- Creation of timing profiles – precise measurement of time spent by different system components to complete various functions; This investigation is an important part of Performance & Load Testing, and also Communication & Interoperability analysis; In certain cases, profiling may be performed in the absence of source code;
- Detailed analysis of reliability and stability – operability testing of different system components whilst in continuous operation; The investigation presupposes full-scale product functionality testing with fixation of source component faults and defects**;**

- Component testing – quality analysis of individual components or structural parts of the system by means of integrated testing of their functionality and interfaces in an artificially created environment; Very often, this investigation includes functionality and performance testing; In many cases, component testing may be performed in the absence of source code;
- Correlation testing – an investigation directed towards the analysis of the mutual effect on each other of different components or program layers; During the analysis process, a series of system components may be replaced by stubs or subjected to independent testing; In many cases, systems based on component architecture, may have correlation testing performed in the absence of source code;
- Template testing – a comparative quality characteristics analysis of individual components or system structural parts (most often performance characteristics) with analogous characteristics to authoritative templates; A template, as a rule, is a simple component or program performing largely analogous functions to the real component. However, it is built into simple architecture and deprived of non-basic business logic, computation and input-output functions, inevitably present in the system; Template characteristics may be obtained by experimental or analytical means;
- Error and defect localization – an investigation directed towards the search of source component problems, detected during software testing; As a rule, we are talking about functionality and performance testing; However, this investigation is equally useful for cause analysis of defects discovered as a result of other investigations; During the course of the investigation, analysis of functionality correspondence between individual system components is performed, as well as debugging of individual system components;
- Static Analysis of Source Code – checking of semantic and syntactical correspondence of software source code according to defined rules; Essentially, this type of investigation is performed in order to control correspondence of system source code to corporate standards, or to detect cases of usage of "forbidden" constructions or functions. The latter, is essential in performing system branch certification or in re-engineering large software systems.

Cause analysis and test results interpretation

AQT experts' high qualifications and vast experience obtained in working on large SQA projects, allows us to present our clients with turn-key solutions to the software quality assurance. As well as test investigations, systematization and appraisal of results, we also perform cause analysis on the appearance of various defects and problems.

Software quality is characterized by a multitude of parameters, the definition of which requires the performance of a large number of tests and analytical results processing. Upon investigation conclusion, AQT presents the client with full information on the characteristics of the product that interest him in the format of a formal report. As well as raw data, the report contains detailed descriptions of work performed, diagrams, graphics, computed metric significances and an experts conclusion, essential for convenient information comprehension.

After this, the software developers face the task of localization and elimination of problems detected during the investigations. Cause Analysis of defects detected and interpretation of measured characteristics with the goal of optimization is a no less labor-intensive process than testing and processing the results. It is essential to complete appraisal of the system architecture and source code, perform debugging of individual program components, and also, possibly, perform additional test investigations. To be at its most efficient, the given process should be completed with the participation of the specialists that discovered the problems.

AQT proposes a more convenient and effective solution: cause analysis of the reasons for defect appearance, test results interpretation and definition of possible methods of eliminating the problems by

our specialists**.** Irrespective of whether we are talking about product functionality, performance or security characteristics, AQT will perform cause analysis on problems detected quickly and effectively.

Part of the AQT team consists of highly qualified professional programmers with a broad grasp of contemporary technologies and development methods. In close cooperation with client specialists, AQT experts will perform an exhaustive analysis of investigation results and present exact information on the reasons for the appearance of problems detected and possible ways to solve them. During the work process, apart from system architecture and source code analysis, as well as profiling and debugging of program components, AQT will perform a series of specialized investigations, significantly increasing the effectiveness of the cause analysis process and the reliability of conclusions reached. Many of these measures are described in detail in White Box Testing & Source Code Analysis.

Regression Testing

The **Rational Unified Process** Software Engineering Methodology regulates the iterative approach to software development. All project stages, namely – planning, analysis, design, development and results evaluation of work completed are performed repeatedly for each stage of work on the project. A full production cycle is performed for practically each step.

The given approach provides a high degree of production systematization, increases the quality of production and significantly decreases risk. The iterative approach is particularly attractive when we are talking about software quality assurance.

On one hand, defects already detected must be corrected and this must be confirmed by corresponding tests in the control process pertaining to the quality of the next software release. On the other hand, the quality of each new release must be confirmed by comprehensive testing of its functionality and other characteristics.

In this way, we have a set of essential tests already formulated, which are to be performed from release to release, regularly controlling software quality. During the process of system development or modification, the test set will change. However, the list of tests to be completed must always provide comprehensive quality control over current system functionality and check correction of defects discovered in previous iterations.

This iterative process is called regression testing. AQT has successful experience in the of IT project quality assurance according to the tenets of this methodology, controlling it from the business process modeling stage to the maintenance stage. The method we use to manage defect database management provides the client with convenient access to information in real time and eases the task of report preparation.

QWE2000 Keynote Session K1-3

Dr. Phillipe Aigrain
(Head of "Software Technologies" Sector, EC, Brussels, Belgium)

A Wider Look at Software Quality

About the Speaker

Dr. Aigrain is Head of Sector "Software Technologies" in the unit "Technologies and Engineering for Software, Systems and Services" of the European Commission Information Society Technologies R&D Programme. He was trained as a mathematician and theoretical computer scientist, and holds a Doctorat and the Habilitation à Diriger les Recherches from University Paris 7. From 1972 to 1981, he worked in software engineering research labs of software companies. He was a research fellow at U.C. Berkeley in 1982. Since then, and before joining the European Commission in 1996, he headed research teams in the field of computer processing, indexing, retrieval and interaction for audiovisual media (video, music, still images). He his the author of more than 60 technical papers, as well as of papers on the economy and sociology of information exchanges.

A wider look at software quality

Philippe Aigrain, IST Programme,
European Commission, DG Information Society

Summary:

The communication proposes to extend the vision of what software quality means, and of possible ways to attain it, in comparison to what the specialised community of software quality experts has usually focused on. It justifies this need by recent trends in the nature and role of software, and difficulty for classical approaches to deal with some important issues. It then describes how recent orientations in the European IST research programme try to deal with these challenges.

Author's details:
Philippe Aigrain, European Commission, INFS0/E2, Office N105 3/54,
rue de la Loi, 200, B-1049 Brussels, Belgium
tel: +32.2.296.0365, fax: +32.2.296.7018, email: Philippe.Aigrain@cec.eu.int

1. Abstract

Several reasons strongly advocate for a re-definition of what software quality means, and how we can try to attain it:

- The growing *ubiquity of software in objects of everyday life* (in contrast to previous situations in which software was found mostly in professional environments),
- The related fact that software is often used for activities that are not clearly known at the time at which it is designed, but on the contrary result from *creative interactive usage* of its functionality,
- The *convergence between computing and telecommunications* , and the resulting complexity, inter-dependency and difficulty for the user to identify the sources of problems in behaviour and performance,
- The fact that a given application or activity supported by software uses components coming from different sources, often *integrated at run-time only*, with similar consequences for users,
- The *emergence of free / open source software* as a major enabler for new ways to attain quality by bridging the gap between users and developers, by making the internals of software visible to a wide population of critical eyes, and by making it possible for alternate developers to correct quality failures,
- The related fact that even for a single piece of software, development is done more and more often by people distributed across different organisations and/or locations.

At a more technical level, Anthony Finkelstein and Jeff Kramer, in their introductory chapter to a book published at the recent ICSE 2000 Conference, (Finkelstein and Kramer, 2000) have summarised the major research challenges of software engineering in a way that takes in account the trends listed above. They suggest to put emphasis on enabling compositionality, adaptation to change in requirements, modelling non-functional properties, shifting to a service view of software, creating new structuring schemes making it possible for software engineering to deal separately with various concerns, adapting to non-classical life-cycles, making it possible to reason about software architectures, enabling user configuration and more generally user development, and exploiting domain specificity.

To justify why these developments and perspectives call for a revised and extended definition of software

quality, the communication first briefly reviews some of the classical approaches to reaching software quality such as:

- Programming constructs and styles,
- Proven behaviour,
- Reliability, fault tolerance, and testing
- Quality from procedures and development processes,
- Capability maturation,
- User-centred design and requirement engineering,

Though the contribution of these approaches to the state-of-the-art must be acknowledged, one also has to recognise their built-in limits in how they can address software quality.

Finally the communication illustrates how recent orientations in the *Technologies and Engineering for Software, Systems and Services* part of the European Information Society Technologies research and technology development programme (IST, 2000) aim at supporting research that extends the approach to software quality, in order to deal with the new challenges of the information society. Recent and planned action lines in our programme address challenges such as:

- Distributed software development
- Engineering of end-user services
- Interaction and functionality design
- Handling the specific needs of free / open source software development
- Models and management of software architecture

References

- (Finkelstein & Kramer, 2000) Anthony Finkelstein and Jeff Kramer, "Software Engineering: A Roadmap", Introductory chapter to "Future of the Software Engineering", ACM Press, 2000.
- (IST, 2000) http://www.cordis.lu/ist

QWE2000 Keynote Session K2-1

Linda Crispin
(iFactor-e)

Stranger in a Strange Land: Bringing QA to a Web Startup

Key Points

- How to get buy-in from management, information systems, development and marketing in a startup environment
- How to educate yourself in testing Web applications
- How to remove the testing bottleneck in Web development

Presentation Abstract

It perhaps goes without saying that a Web startup is not an environment in which quality testing is typically found. Development is fast and loose. Many developers are inexperienced. They're racing to be first to market. One might be tempted to label the environment as chaotic.

When I accepted the opportunity of being the first test engineer at TRIP.com, only 25 people worked for the Web startup. The developers had produced some exciting applications and felt they were ready to "grow up and play with the big boys." The development team thought they were intellectually prepared to introduce standards and procedures.

In reality, development was frenetic, and the developers didn't have a clue as to how to stop and analyze their processes, much less how to impose discipline on them.

For my part, I was a complete stranger to Web development. For years I had been testing databases, 4-GLs, and client/server software on UNIX, NT, and Windows platforms. I spoke ODBC, but not JDBC. I knew my customers. In my experience, the software development cycle had stretched on for months or even years-during which your typical Web application has gone though numerous incarnations.

This is the story of how I learned about Web application development, preached the quality gospel, and collaborated with the software and product developers and marketing managers to implement development standards and project processes that build quality into our applications. TRIP.com now employs 200 people, has three million registered customers, and has introduced such cutting-edge products such as intelliTRIP and companyTRIP.

About the Speaker

I have eighteen years experience in the industry with the last nine in Testing and Quality Assurance. I started out as a programmer and later worked in customer support and QA for large software vendors. In March of 1998, I discovered the world of Web startups, joining TRIP.com as the first test engineer. The challenge of building quality into Web applications while meeting tight development cycles was eye-opening. At TRIP.com, I built a QA department of seven test engineers testing state-of-the-art, first-of-their-kind applications such as flightTracker and intelliTRIP. I felt, however, that we never found a really good process that worked to produce high-quality software in a short amount of time. Missed deadlines were common. Still, we were proud of our accomplishments, as TRIP.com grew to one of the highest-traffic travel Web sites, rated 4.5 out of 5 starts by BizRate and ranked near the top of the Keynote Top 40 websites for performance.

Several developers from TRIP.com left to join a new startup, iFactor-e, devoted to using eXtreme Programming to combine high quality and short time to market to wow the customer. One of these developers loaned me Kent Beck's book. After I read that, I was eager to try XP myself and was fortunate enough to be hired as the first test engineer at iFactor-e in July of 2000. Since then, I have been racing to establish a functional testing methodology that successfully applies the values of XP.

I have given successful presentations at both local and international user and QA conferences to audiences of up to 60 people. I have many years experience training both technical and end users.

Stranger in a Strange Land: Bringing QA to a Web Startup

Lisa Crispin
Senior Test Engineer
iFactor-e

1

Bringing QA to a Web Startup

What I found in the DotCom environment

- Frenetic pace
- Lack of experience in process, standards
- Little comprehension of quality and testing

2

Bringing QA to a Web Startup

Who I was when I started at a DotCom

- No experience in web testing
- Used to long development cycles
- Used to a corporate environment with lots of process

3

Bringing QA to a Web Startup

This Presentation Will Give Tips On:

- How to get buy-in
- How to educate yourself
- How to prevent a testing bottleneck
- How to define a sane process
- How to create documentation

4

Bringing QA to a Web Startup

Get Buy-In

- Identify a top manager to champion your cause
  - Who understands your mission
  - In leadership position, eg. VP of Development

5

Bringing QA to a Web Startup

Get Buy-In

- Partner with someone outside of your group
  - Eg., project or operations manager
  - Educate them and garner their help

6

Bringing QA to a Web Startup

Get Buy-In

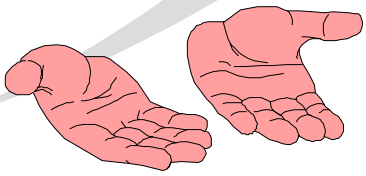
- Educate everyone about software quality assurance
  - Eg. - hold professional development session
  - What QA will do for them and the site
  - Meetings, face time

7

Bringing QA to a Web Startup

Get Buy-In

- Support Information Systems
  - Don't bug them for what you can do yourself
  - Let them do their job
  - They hold the keys and are on your team

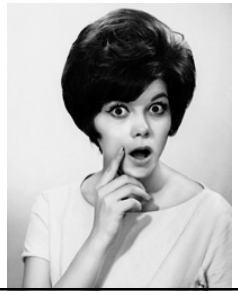


8

Bringing QA to a Web Startup

Get Buy-In

- Listen to the developers
  - Try to understand their point of view
  - They may communicate differently
  - Quality Hero award



9

Bringing QA to a Web Startup

Get Buy-In

- Brainstorm
  - Developers, Database IS, Marketing, Customer Support
  - Potentially overlooked problems

10

Bringing QA to a Web Startup

Work Smart

- Evaluate and select appropriate tools
  - Test, defect tracking, configuration mgmt
  - Vendors who can help you
  - Consider learning and implementation time
  - Doesn't have to be expensive or famous

11

Bringing QA to a Web Startup

Work Smart

- Search the Web for resources
- http://www.softwareqatest.com/index.html
- http://www.kaner.com/writing.htm
- http://www.crl.com/~zaller/testing.html#Kerryh
- ttp://www.testingcraft.com/cgi-bin/wiki.cgi?FrontPage

12

Bringing QA to a Web Startup

Work Smart

- Hire good help
  - Senior, experienced testers
  - Or train your own - attitude is key
  - Spend the $

3

Bringing QA to a Web Startup

Work Smart

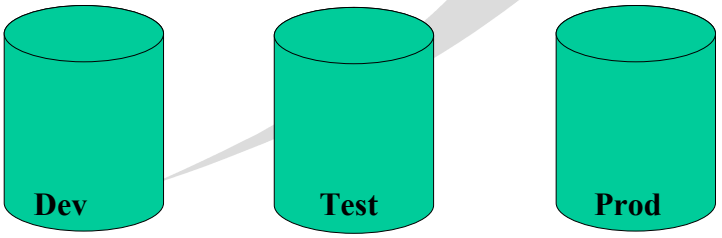
- Get input about quality from everyone
  - Form a quality board
  - Hold quality review panels

14

Bringing QA to a Web Startup

Work Smart

- Insist on a dedicated test environment
  - Looks like production
  - Can handle production level load test

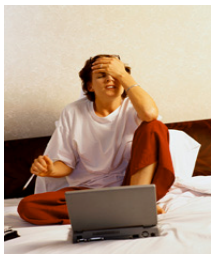


15

Bringing QA to a Web Startup

Work Smart

- Learn about the development tools
  - Quality issues - version control, workflow
  - Configuration and performance issues
  - Ask questions

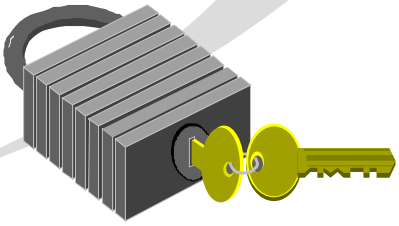


16

Bringing QA to a Web Startup

Define Process

- Get control of production environments
  - Lock developers out!
  - Work with Information Systems



17

Bringing QA to a Web Startup

Define Process

- Get involved from the beginning
  - Review all documentation
  - Agree on one vision for each project
  - Define QA and Dev process
  - Hire a good tech writer

18

Bringing QA to a Web Startup

Define Process

- Innovate!
  - New documentation formats
  - Consider alternatives such as Extreme Programming

19

Stranger in a Strange Land:
Bringing QA To a Web Startup

Lisa Crispin, Senior Test Engineer, iFactor-e

It perhaps goes without saying that a Web startup is not an environment in which quality testing is typically found. Development is fast and loose. Many developers are inexperienced. Marketing is pushing to be first to market. One might be tempted to label the environment as chaotic.

When I accepted the opportunity of being the first test engineer at TRIP.com, only 25 people worked for the Web startup. The developers had produced some exciting applications and felt they were ready to "grow up and play with the big boys." The development team thought they were intellectually prepared to introduce standards and procedures.

In reality, development was frenetic, and the developers didn't have a clue as to how to stop and analyze their processes, much less how to impose discipline on them.

For my part, I was a complete stranger to Web development. I truly felt lost in the wilderness without a map or a compass. For years I had been testing databases, fourth-generation languages, and client/server software on UNIX, NT, and Windows platforms. I spoke ODBC, but not JDBC. I knew my customers. In my experience, the software development cycle had stretched on for months or even years-during which your typical Web application has gone through numerous incarnations.

This is the story of how I learned about Web application development, preached the quality gospel, and collaborated with the software and product developers and marketing managers to implement development standards and project processes that build quality into our applications. TRIP.com now employs over two hundred people, has over four million registered customers, and has introduced such cutting-edge products such as FlightTracker and intelliTRIP. Myself, I've moved on to a new startup. Once again, I feel like a stranger - this time in the strange land of eXtreme Programming. But that's another paper.

Any DotCom is a work in progress. Even once we had a successful project process at TRIP.com, we faced continual challenges such as an inadequate test environment. Even when the whole company understands and is committed to the importance of quality assurance testing, unexpected events lead to surprises. The key is to keep plugging away at the following tasks:

- Get Buy-In
- Work Smart
- Define Processes

I. Get Buy-In

I won over my managers, developers, and marketing counterparts by following these tenets:

- **Identify** a top manager in your organization who believes in your cause and will champion it. In my case, it was the Vice President of Web Development and Chief Cat Herder (yes, that really was her title). When I was hired, this person did not believe that five developers could keep one tester busy full time. But, in time, she became my biggest ally. She not only pushed the developers to work for quality, but she also lobbied the management team for testing resources.
- **Partner** with someone outside of your organization, such as a project or operations manager. Educate your ally to garner his or her help. We had a topnotch project manager who, once she understood what QA and

testing would do for the company, did much of my job for me. She enforced processes such as document review and signoff, helped implement and police the defect tracking system, and tied up a million details involved with every big production launch. She became the prime channel of communication between marketing and development.

- **Educate** everyone you come into contact with at the company about software quality assurance: what it is and what it will do for them and the site. Early on, I held a "Lunch 'n' learn" professional development seminar. The company bought lunch, so attendance was good. I explained why testing is essential and what it involves. Lots of meetings and one-on-one encounters are needed to get everyone on board and to establish priorities.
- **Support** Information Systems, the group that administers the production site. These people will benefit from not having their pager go off so much when applications are tested before being launched to production. The Vice President of Technology and the IS director fully supported me and refused to launch any update that had not been fully tested. This kept the rest of the organization from steamrolling over me, giving me a chance to prove the value of testing. I don't bug IS for the things I can do myself, but I let them do their job. For example, I installed Y2K patches on the test machines, but IS controlled all the UNIX and NT account management.
- **Understand** the developers' point of view. You may have young, brilliant developers who don't communicate in ways you are accustomed to. Our original developers were mostly very young and inexperienced. Some had not finished high school!. Others weren't old enough to drink! The culture was anti-corporate, and they said what was on their minds. I found that if I listened, I learned, and they in turn were willing to listen to my ideas. I learned everything I now know about Web applications from the developers themselves. I presented a "Quality Hero" award each month to a developer who took exceptional measures to prevent defects and improve quality. The prize was just a Nerf gun and the developer's name appeared on the Quality Hero Award plaque, but it raised the visibility of high quality process and techniques.
- **Brainstorm** with developers and others about problems that may not come out in testing. For example, I didn't know that if you change a URL, search engines may not be able to find your site. When we implemented a content management tool that required changing every URL, this was important information!

II. Work Smart

Here's my advice for making the testing organization lean and mean:

- **Evaluate tools**. Put as much time as you can into tool evaluation, such as those for automated testing, defect tracking, and configuration management. Identify the vendors who can help you the most, and get as much information from them as you can. Ask fellow testers for their recommendations and experiences. Install new tools and try them out. Select tools that are appropriate for you and your company. It doesn't do any good to buy a tool you don't have time to learn how to use, especially if your testing team is small. I ended up choosing tools that are lesser known but stil meet our needs. For example:
  - For automated testing at TRIP.com (we use this at iFactor-e as well), we used WebART, an incredibly inexpensive, easy-to-learn , but powerful tool sold by OCLC Inc. (a non-profit company). They gave me invaluable advice and provided insights about Web testing. Pick everyone's brain including vendors!
  - For defect tracking at TRIP.com, we chose a Web-based tool, TeamTrack (now called TTrack), from a startup company called TeamShare . It, also was far less expensive than its competitors, but it was easy to implement and customize. At iFactor-e, which is a brand-new startup on a small budget, we use the free Bugzilla and it is fine for our needs.
  - For configuration management, we again turned to a smaller, innovative company which produces an inexpensive, easy to implement and learn yet robust tool, Perforce. At iFactor-e, we use freeware, CVS - it lacks some features, but the development team is small and can work around its drawbacks.

These tools won't necessarily meet *your* needs - just be open and creative when evaluating tools. Investigate alternatives - for example, if you create unit tests to reproduce each bug you find in testing, you may not even need a defect tracking system!

- **Search the Web for resources**. I would have quit TRIP.com after a week if I had not found an excellent Web site that points to information about testing Web applications and lists of tools. These sites, in turn, led me to more tools and information Here are some examples:

  http://www.softwareqatest.com/index.html
  Everything from basic definition and articles on how to test Web applications to comprehensive lists of Web tools to links to other informative sites.
  http://www.kaner.com/writing.htm
  Articles by Cem Kaner
  http://www.crl.com/~zallar/testing.html#Kerry
  l
  ong lists of associations, vendors, tools, training, reference information, conferences, interesting papers.
  http://www.testingcraft.com/cgi-bin/wiki.cgi?FrontPage
  a Wiki forum for exchanging test techniques and the related
  http://www.egroups.com/group/testingcraft-discuss

Of course, user conferences are invaluable for both information-gathering and networking.

- **Hire good help**. It proved impossible at first to find experienced test engineers in our area, so we at TRIP.com hired bright but inexperienced people with the right qualities that make good test engineers: enthusiasm, dedication to the end user, and determination. A caveat: inexperienced testers who have no programming experience have a harder time learning a scripting language for an automated test tool. However, by using a combination of outside classes, hiring consultants and patient, one-on-one training, our testers learned UNIX, SQL, test scripting, HTML, configuration management, and other technical skills. You're going to spend money either way - paying high salaries for experienced test engineers, or training novice testers. Once you've turned them into pros, remember to keep them challenged, happy and well-compensated so other companies don't poach them!
- **Get input** about quality from all departments in the company. I formed a quality board with members from sales, marketing, customer support and travel to gather fresh ideas about error prevention and prioritization of regression testing. Whenever major problems occured, we held a quality review panel where representatives from development and information systems heard short presentations from the people who experienced and fixed the problem. The panel studied the issues and recommended steps to prevent such problems from recurring. This was a big effort and to be truthful, it was difficult to get follow-through. But give it a try. We also held post-mortems after all major launches to see what lessons could be learned. By employing these methods, we learned some valuable lessons!
- **Insist on a test environment** that is exact replica of, but is entirely independent from, production. You can't emulate a production load without the equivalent of production hardware and software. Since the production architecture is likely to change in response to increased traffic and other considerations, this is a moving target. The test environment will need to be updated in synch with the production environment. The architecture is key too. If the production servers are clustered, your testing had better be done in a clustered environment. If part of an application runs on a stand-alone machine, it must do so in your test environment. Establishing and keeping up development, test and production environments was a huge challenge. Even when the entire company is sold on the idea of a proper test environment, there are business and technical reasons (read: excuses) that get in the way of reproducing the production environment in for testing . Don't be complacent, and never give up. Make sure you have the best test environment you can get for each application going into production, and work actively with your information systems team to get the environment you really need. Even small applications can deceive you. For example, we launched a simple application, SantaTracker, on Christmas Eve so kiddies

could watch Santa's sleigh fly around the country. The test environment was broken, so we were not able to test on the same architecture that it was to run in production, but we weren't concerned. After all, it should have worked exactly like our regular FlightTracker application! Right? Wrong! It was a disaster!

In short: Dig your heels in and refuse to launch until some semblance of a test environment is established. Remember, it is harder to get the test environment once the new application is in production. Make it a requirement of release.

- **Learn about the development tools.** They present their own quality issues and offer some solutions, too. For example, the first version of our content management tool did not have any version control. It took more than a year to upgrade to the release that offered this capability, and even then it didn't ***enforce*** version control. We had to constantly police the process to ensure that developers version their code. The software on which our Java applications were based had complex configuration parameters we didn't fully understand when we first put it into production. We had tested our intelliTRIP product with a production load in terms of transactions per second, but never with a realistic number of concurrent users. As a result, the servers kept crashing on the first day we put it in production. If we had understood the configuration and the user session management parameters better in our Java-application management tool better, we could have prevented this problem.

III. Define Processes

Collaborate with your counterparts to formalize your processes:

- **Get control** of the production environments. Work with your information systems team to create a production update procedure. When I started, developers launched their own changes to production. It was hard to wean them away from this bad habit. Even after we thought we had implemented good production update procedures, we lacked the discipline to enforce it under pressure. For example, since we did not have good configuration management, it became impossible to build baselines of intelliTRIP, so developers would simply move new classes into production to fix problems. Only?after two years did ?we begin to be able to require developers to build scripts and installation documentation before we accepted any software from development for testing.
- **Get involved** from the beginning of each project. This is hard work. It forces you to juggle many tasks, but it is essential. Participate in all documentation reviews: Those for requirements, functional specifications, and design specifications. Make sure the documents are complete and clear. Look for ambiguities, gaps, lack of detail. All parties, including marketing, development, test, customer support, sales must agree on a vision for the product. This vision is a short phrase that describes the main thrust of the product. With intelliTRIP, development and test were told to produce a server-side version of the original client-side product as quickly as possible. Sales and marketing believed that the purpose of the product was to ***quickly*** locate fares from airline Websites. Since development and test wasn't told that part about ***quickly,*** we released the product even though we knew that is was sometimes slow to return results. This type of disconnect can be prevented by including a vision statement in the requirements
- **Define quality**. Work with marketing and product development to define quality IS for each product.: Should the priority be good, fast, or cheap? (You can only have two of the three!) Even if you choose fast, don't sacrifice the process. At TRIP.com, we once implemented a promotion that marketing believed to be simple and wanted to rush to production. Since the product manager did not hold documentation reviews and get signoff, the HTML pages produced by the developers had to be changed three times. This took much longer than a documentation review meeting. There is no need to get bogged down in process either. If you find that is happening, change the process, or train people how to use it properly.
- **Document** the internal processes of both test and development. You can't expect marketing and product development to follow best practices if you don't do it yourself. At TRIP.com, one of the development directors, with the help of the technical writer, led an effort to define and document the development

process. By the way—?no matter the size of the company, unless you are using a lightweight technology such as eXtreme Programming, you need at least one experienced, skilled technical writer in your development organization.

- **Enforce the process.** If your QA team is large enough, dedicate one person to administering and enforcing configuration management and delivery of installation scripts and documentation. At TRIP.com, we expanded this Configuration Manager role to that of a deployment engineer who works closely with developers to produce the builds and installation procedures. Don't accept software to test if it is not accompanied by all the documentation and software you need to promote, test, and launch it to production
- **Innovate!** Look for new ways to present documentation such as functional and specifications. Web applications require a new approach. You have creative people at your company who can help! Get input from as many different groups as you can. If your company really wants to get innovative, consider lightweight methodologies such as eXtreme Programming (XP). These methodologies can meet the need to get to market quickly, accomodate rapidly changing requirements but still release a high quality product.

Summary - As You Grow

All companies change as they grow beyond the 'startup' size and environment. My team and I endured countless frustrations when fast growth at TRIP.com led to temporary chaos. As your organization grows, educate new employees about project process and quality practices. Listen to them and take advantage of their fresh outlook and new ideas. Take the initiative. If a gap results from a re-organization, fill it yourself. For example, when we were temporarily without a project management function in the company, the development director and I set up a weekly tactical meeting with representatives from all departments so that everyone could stay informed and juggle resources. Quality assurance can be a frustrating job, especially in a Web startup. Pick your battles. Keep striving for better quality. Above all, enjoy the experience!

QWE2000 Keynote Session K2-2

Mr. Hans Buwalda [Netherlands]

(CMG Finance BV)

"Soap Opera Testing"

Key Points

- Business oriented approach to test development
- Pitfalls of the traditional mechanical ways to develop tests
- Practical ways to apply scenarios as an alternative

Presentation Abstract

It is far from easy to develop good tests. Translating requirements one by one into test cases isn't always good enough. You can end up with an unmanageable volume of boring tests, that lack effectiveness in finding complex and hidden problems. Based on experiences in numerous projects around testing and test automation a number of techniques will be presented that can improve the process of test development. They are grouped around the theme "Soap Opera Testing". Applying them can not only lead to a better manageable test set, it is also a more motivating and creative way of developing tests.

The idea behind soap opera's is that you use real business examples for your tests in the form of scenario's. But, comparable to soap opera's on television, they are condensed and, if appropriate, exaggerated. In this way you can find out if a system can cope with complex business situation and, even more important, it motivates end users to make nice test cases (it is more fun).

This theme has been presented in the States as keynote for Star East 2000 and has raised very positive evaluations from the audience.

About the Speaker

Hans Buwalda is project director at CMG, a leading European information technology services group. He is responsible for new developments around the TestFrame approach for testing and test automation of which he is the main architect. The approach has been started by him in 1994. In 1996 he presented the main ideas for the first time to an international audience in a speech called "Testing with Action Words, abandoning record and playback". Since then the method is being used in an increasing number of countries and Hans has become a frequent speaker at industry conferences, tutorials, and workshops.

CMG

International Software Quality Week Europe 2000

Brussels BELGIUM 20-24 Nov 2000

Soap Opera Testing

TESTFRAME

Hans Buwalda
CMG - TestFrame Research Centre
hans.buwalda@cmg.nl
www.testframe.com

CMG

TESTFRAME

Agenda

- introduction
- underlying architecture: the TestFrame Model
- Soap Opera's
- usage

CMG TESTFRAME

The Challenges for a Test Process

- testing should be fun
- testing should be effective
- testing should be efficient
- testing should be under control

CMG TESTFRAME

The “mechanical approach” for test development (example)

- start with (preferably long) list of requirements
- make a test case for every requirement
- use a standardized test technique to translate the requirements into the test cases
- hire (many) people to peform the tests by hand
- ….

CMG TESTFRAME

Some pitfalls with a too mechanical approach

- no fun at all
- inhibiting creativity
- coverage is focussed at single requirement level
- any defects should probably have been found in an earlier test
- suggests false sense of control
- testset hard to maintain
- doesn't catch mistakes in the requirements
- . . .

CMG TESTFRAME

Questions to answer with a test collection

1. does the system comply to the requirements
2. are there any problems (defects and/or failures) we should know about
3. will the system work in practice

CMG TESTFRAME

Re-usable test products in Testframe

A	B	C	D
. . .			
transfer	Houston	Klein	210
check balance	Klein	210	
transfer	Savy	Klein	150
check balance	Klein	360	
. . .			

test development

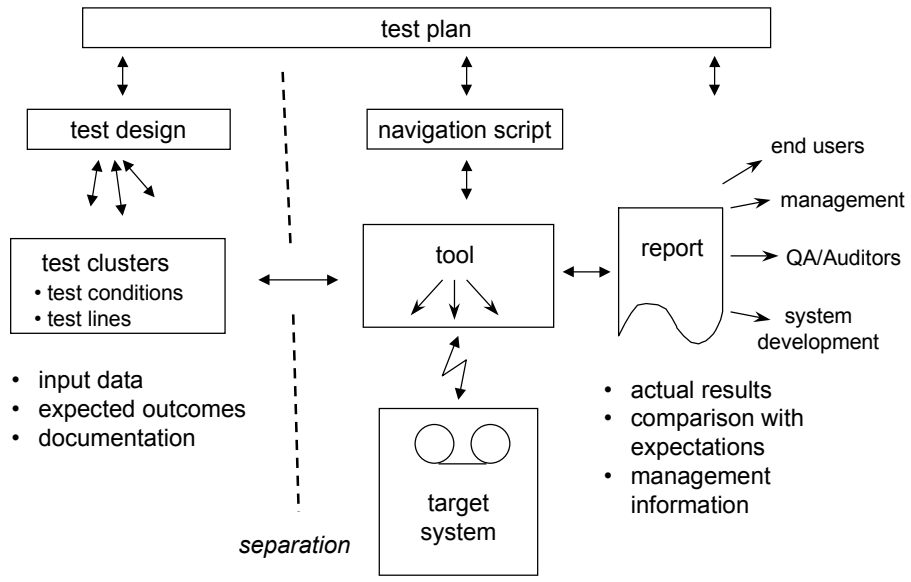
navigation scheme

test cluster

test execution

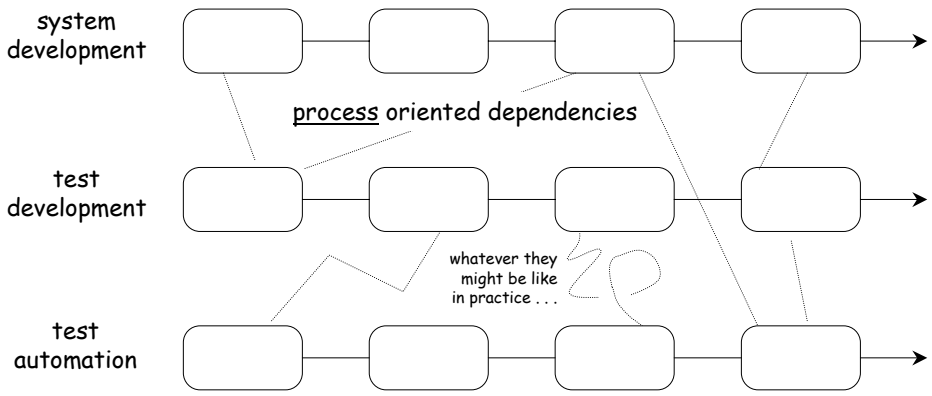
…	
check balance	
enter customer	
…	

CMG TESTFRAME



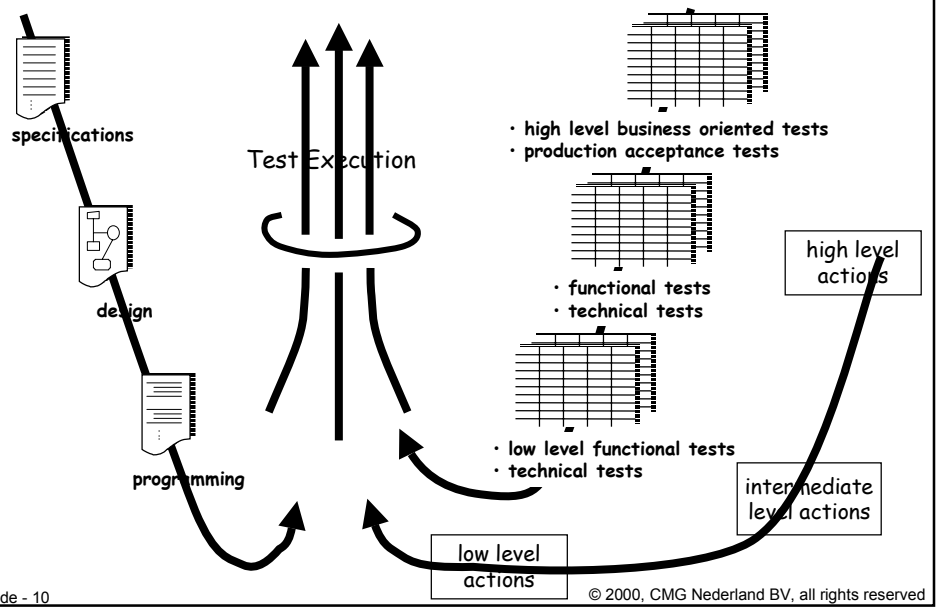
CMG TESTFRAME

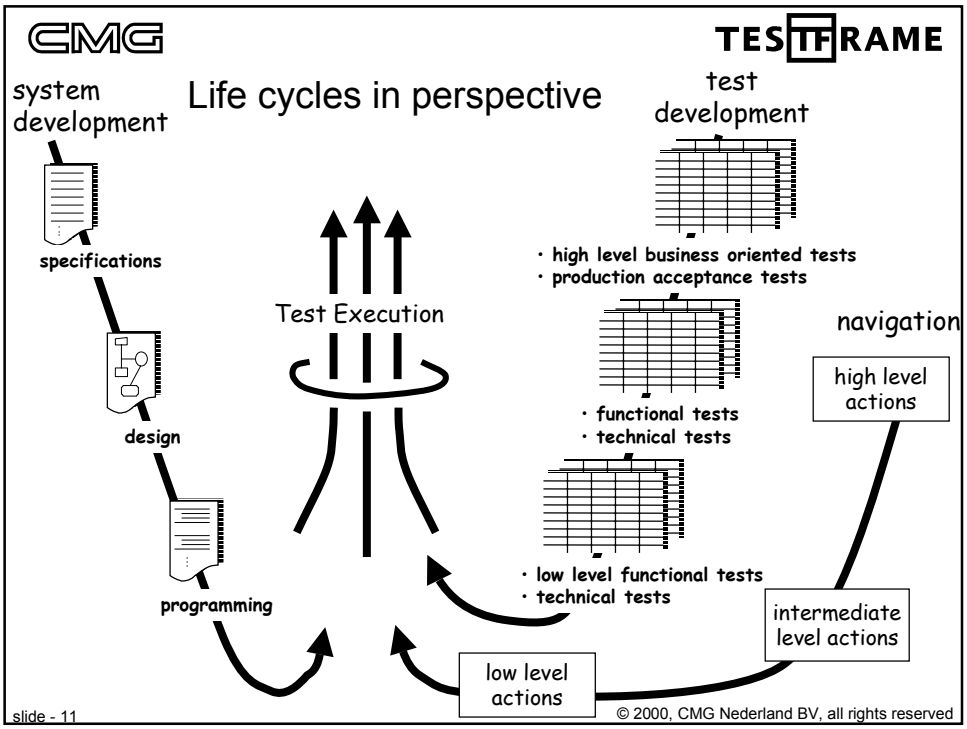
Independence of life cycles

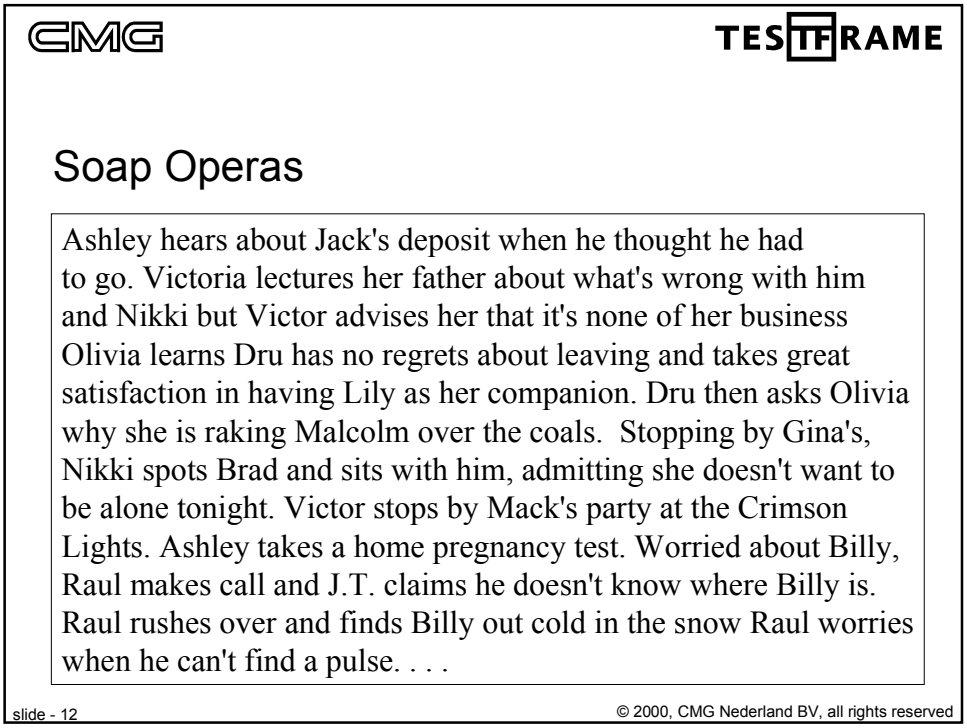


CMG TESTFRAME

Life cycles in perspective




CMG
TESTFRAME
Life cycles in perspective
system development
specifications
design
programming
Test Execution
test development
· high level business oriented tests
· production acceptance tests
· functional tests
· technical tests
· low level functional tests
· technical tests
navigation
high level actions
intermediate level actions
low level actions
slide - 11
© 2000, CMG Nederland BV, all rights reserved


CMG
TESTFRAME
Soap Operas
Ashley hears about Jack's deposit when he thought he had to go. Victoria lectures her father about what's wrong with him and Nikki but Victor advises her that it's none of her business Olivia learns Dru has no regrets about leaving and takes great satisfaction in having Lily as her companion. Dru then asks Olivia why she is raking Malcolm over the coals. Stopping by Gina's, Nikki spots Brad and sits with him, admitting she doesn't want to be alone tonight. Victor stops by Mack's party at the Crimson Lights. Ashley takes a home pregnancy test. Worried about Billy, Raul makes call and J.T. claims he doesn't know where Billy is. Raul rushes over and finds Billy out cold in the snow Raul worries when he can't find a pulse. . . .
slide - 12
© 2000, CMG Nederland BV, all rights reserved

CMG TESTFRAME

Properties of Soap Operas

- about “real life”
- but condensed
- and more extreme

CMG TESTFRAME

Soap Operas for testing

- define a scope of the test to develop
- identify with the business environment
- which elements would make things difficult
- draft scenario’s (typical some dozen lines)
- write them down in clusters

CMG TESTFRAME

Examples of story lines when used for testing

Pension Fund

William starts as a metal worker for Industrial Entropy Incorporated in 1955. During his career he becomes ill, works part time, marries, divorces, marries again, gets 3 children, one of which dies, then his wife dies and he marries again and gets 2 more children….

World Wide Transaction System for an international Bank

A fish trade company in Japan makes a payment to a vendor on Iceland. It should have been a payment in Icelandic Kronur, but it was done in Yen instead. The error is discovered after 9 days and the payment is revised and corrected, however, the interest calculation (value dating)…

CMG TESTFRAME

Example of test lines

	from	to	amount	valuta	trans nr
enter payment	123421344	4124244123	120000	yen	&keep tx1
check value dating	&tx1	$0.47			
wait days	9				
order to reverse	&tx1				
	from	to	amount	valuta	trans nr
enter payment	123421344	4124244123	1200000000	IKr	&keep tx2
check value dating	&tx2	$7,701.56			
. . . .					

CMG TESTFRAME

Soap Operas (in testing) are not necessarily:

- “extreme”
- far fetched
- long and elaborate
- pieces of art and creativity

CMG TESTFRAME

“Killer Soaps”

- more specifically aimed at finding hidden problems
- run when everything else has passed
- one option: put a killer soap at the end of a normal cluster
- ask the “specialists” for input

CMG TESTFRAME

Reasons for scenarios like soaps

- test collection can be made more compact
- it is more fun to make
- specialists used more effectively
- testing more of the application
- less directly dependent on functional specs, so catching more pitfalls

CMG TESTFRAME

What to use it for

primary use

- high level functional acceptance testing

but also:

- module testing
- system testing
- integration testing
- ...

CMG TESTFRAME

What is not interesting for soaps

- screen stuff
- routine tests
- any other straight forward compliance testing

CMG TESTFRAME

Relation to use cases

- friendly cousins
- soap opera's are more directly aimed at testing, for example by exaggerating and using (non local) combinations
- less analytical "top down", but "outside in": translated from end user practice

CMG TESTFRAME

Who could make Soaps

- nearly everybody:
  - end users
  - specialized testers
  - developers
  - auditors
  - ...

CMG TESTFRAME

Ways to get them

- coaching end users or business specialists
- interviews
- own fantasy
- workbooks
- using joint development sessions

CMG TESTFRAME

Joint Testware Development (JTD) ™

- JTD = A technique for structured test development without (complete) documentation
- Based on knowledge of experts instead of just documentation
- JTD <> “Just talk to the end-users and write down what they say”

CMG TESTFRAME

What can joint sessions give you

- Test Strategy
- Acceptance Criteria
- Cluster Grouping
- Test conditions
- Evaluation of Results
- Starting up development of scenarios

CMG TESTFRAME

Joint sessions

- moderator / chairman
- users
- business specialists
- developers
- testers

CMG TESTFRAME

Setup of a joint session for a telecom provider

- 1st session
  - Introduction by moderator and project manager
  - explanation about the JTD procedure
  - explanation of the functional area by a specialized user
- 2nd session
  - start of production of test conditions
- 3rd session
  - start of production of test scenarios
- 4th session
  - evaluation test scenarios

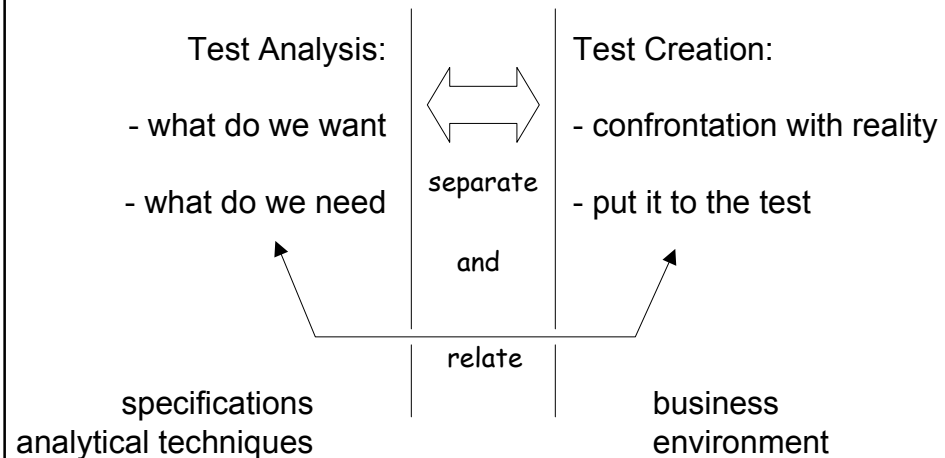
CMG TESTFRAME

Structured test development and Soaps

- soaps are not the natural way to get “coverage”
- additional techniques can help, examples in TestFrame:
  - test conditions
  - test design templates
- recommendation: do “matching” afterwards

CMG TESTFRAME

Test analysis and test creation



CMG TESTFRAME

Example Test Condition

nr	description
...	
3.51	it is checked that the exit date is after the entry date
...	

coupling with the actual test lines in the cluster

test condition	3.51		
	name	entry date	exit date
enter employment	Bill Goodfellow	1999-10-02	1999-10-01
check error message	The exit date must be after the entry date.		

CMG TESTFRAME

Matching Conditions

condition	description	severity	tested in scenario:	
MBT-C01	a customer entered in the enter client screen is present in the database	high	MB01	Entering customers using manual numbering
			MB02	Automatic account numbering
MBT-C02	a customer with a positive balance can transfer money to another customer	high	MB01	Entering customers using manual numbering
			MB02	Automatic account numbering
MBT-C03	a customer with a negative balance cannot transfer money to another customer			
MBT-C04	the balance of the paying customer is decreased with the sum payed	high	MB01	Entering customers using manual numbering
			MB02	Automatic account numbering
MBT-C05	the balance of the receiving customer is increased with the sum payed	high	MB01	Entering customers using manual numbering
			MB02	Automatic account numbering
MBT-C06	account numbers can be entered manually by the user	medium	MB01	Entering customers using manual numbering
MBT-C07	account numbers can be generated automatically by the system	medium	MB02	Automatic account numbering
MBT-C08	...			
...				

CMG TESTFRAME

Test Design Templates

- spreadsheet based technique for designing tests
- introduced in 1998 as an extension to TestFrame by Edward Kit
- friendly technique and yet very strong, for example in identifying needed situations and combinations
- for a further description please have a look at Ed's article in Software Development Magazine:

  http://www.sdmagazine.com/breakrm/features/s992f2.shtml

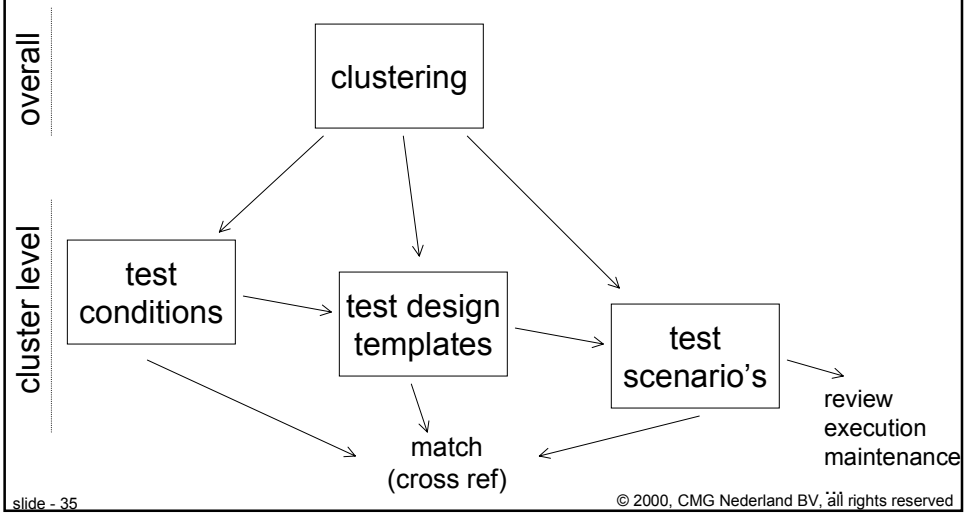
CMG TESTFRAME

Example of Test Design Templates

Template ID:	MB des 1	MB des 2	MB des 3	MB des 4
		….		
customer	*	*	*	*
last name				
first name				
balance	positive	too low	positive	positive
number				
confirmation letter	yes	yes		
automatic numbering				yes
matching		….		
tested in scenario:	MB01		MB01, MB02	MB02

CMG TESTFRAME

A life cycle for test development



CMG TESTFRAME

Questions to answer with a test collection

1. does the system comply to the requirements
2. are their any problems (defects and/or failures) we should know about
3. will the system work in practice

CMG TESTFRAME

Questions to answer with a test collection

		mechanical	soaps	soaps + techniques
1	does the system comply to the requirements	***	*	**
2	are their any problems (defects and/or failures) we should know about	*	***	***
3	will the system work in practice	**	**	***

CMG TESTFRAME

The three “holy grails” of Test Development


effective “clustering” of tests


choosing the right approach per cluster


the proper level of actions

CMG TESTFRAME

Experiences

- the approach can work well once people are used to it
- it can be hard to get away from traditional functional testing
- not magic, use it where applicable (grail 2)
- good clustering is essential (grail 1)
- start with a coach who understands and believes

CMG TESTFRAME

Acknowledgements

- Edward Kit
- Cem Kaner
- James Bach
- Hans Schaefer
- Dennis Janssen
- Chris Schotanus

QWE2000 Session 6T

Mr. Francesco Piazza [Italy]
(Alenia Aerospazio)

"A Requirements Trace Application"

Key Points

- Practical and usefull simple database application
- Industrial experience on trace requirements subject
- Standard approach on document formatting to make easier requirements tracing management

Presentation Abstract

The present abstract is aimed at describing a company methodology for requirements tracing. The methodology foresees some working rules and is based on an set of conventions and on an application built using the ACCESS '97 relational database by Microsoft. The methodology has been widely used in ground and flight satellite systems built by the company.

About the Speaker

Mr. Francesco Piazza graduated in Computer science with 106/110 at the University of Pisa in 1979 with a thesis on concurrent programming. HeÆs been working in ALENIA AEROSPAZIO S.p.A. since 1990 in engineering department and since 1994 in the Central Quality Direction. Now he is working on Information Technology Center. He has worked for several years in other companies in the following areas: system qualification tests, graphic computing, software tools.

A REQUIREMENTS TRACE APPLICATION

Summary:
A Requirements Trace Application, based on a database application, is described herein. Details on methodology, requirements document structure and organisation, man machine interface, implementation issues are given in the paper.

Francesco Piazza, Dr., Alenia Spazio Company
ph: +39-06-41512244 fax: +39-06-41512433
e-mail: f.piazza@roma.alespazio.it

Keywords: Requirement, Traceability, Software, System, Subsystem, Test

1 Introduction

The present paper is aimed at describing the experience of Alenia Aerospazio in the "in house" development of a Requirements Trace Methodology (RTM), using an application based on a database management system supporting the relational data model.
Both the methodology and the application have been widely used to support the verification and validation of ground and flight satellite systems developed by the company.

The methodology allows a structured requirements management during design and test phases from high levels toward bottom. It allows a strict control on requirements distribution among different working groups. Impacts of changes in requirements can be better evaluated and controlled during the projects development. The methodology foresees some working rules and is based on a limited set of conventions.

The methodology covers requirements tracing from *system requirements* specification toward subsystem, software requirements, architectural design elements; moreover it covers tracing among requirements and related test specifications (see fig. 1 for the documents relationships, and for the related logical database schema). Both forward and backward requirements tracing have been fully implemented.

The factors driving the decision to develop in house a methodology and the corresponding application have been:

- the growing number of requirements imposed on a space system, often hierarchically layered and decomposed (from system requirements, to S/S requirements down to software requirements;
- Test steps and involved environments in the overall testing process are complex, spread on different

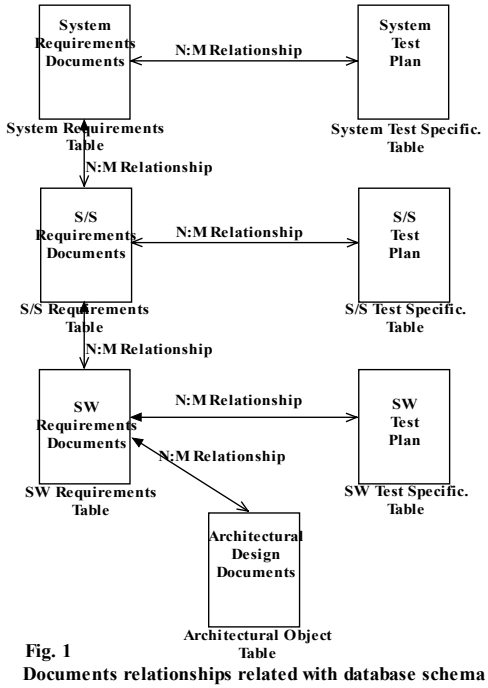


Fig. 1
Documents relationships related with database schema

phases and with different aims, using different platforms;
- Higher flexibility reachable with a totally internally controlled application. This flexibility allowed to adapt the application to different scenarios ranging from flight to ground segment requirements specifications. The flexibility allowed to better cope with customer requirements.

The experience has been greatly influenced by the joined work with the effort of the system designers that required support in managing requirements tracing.

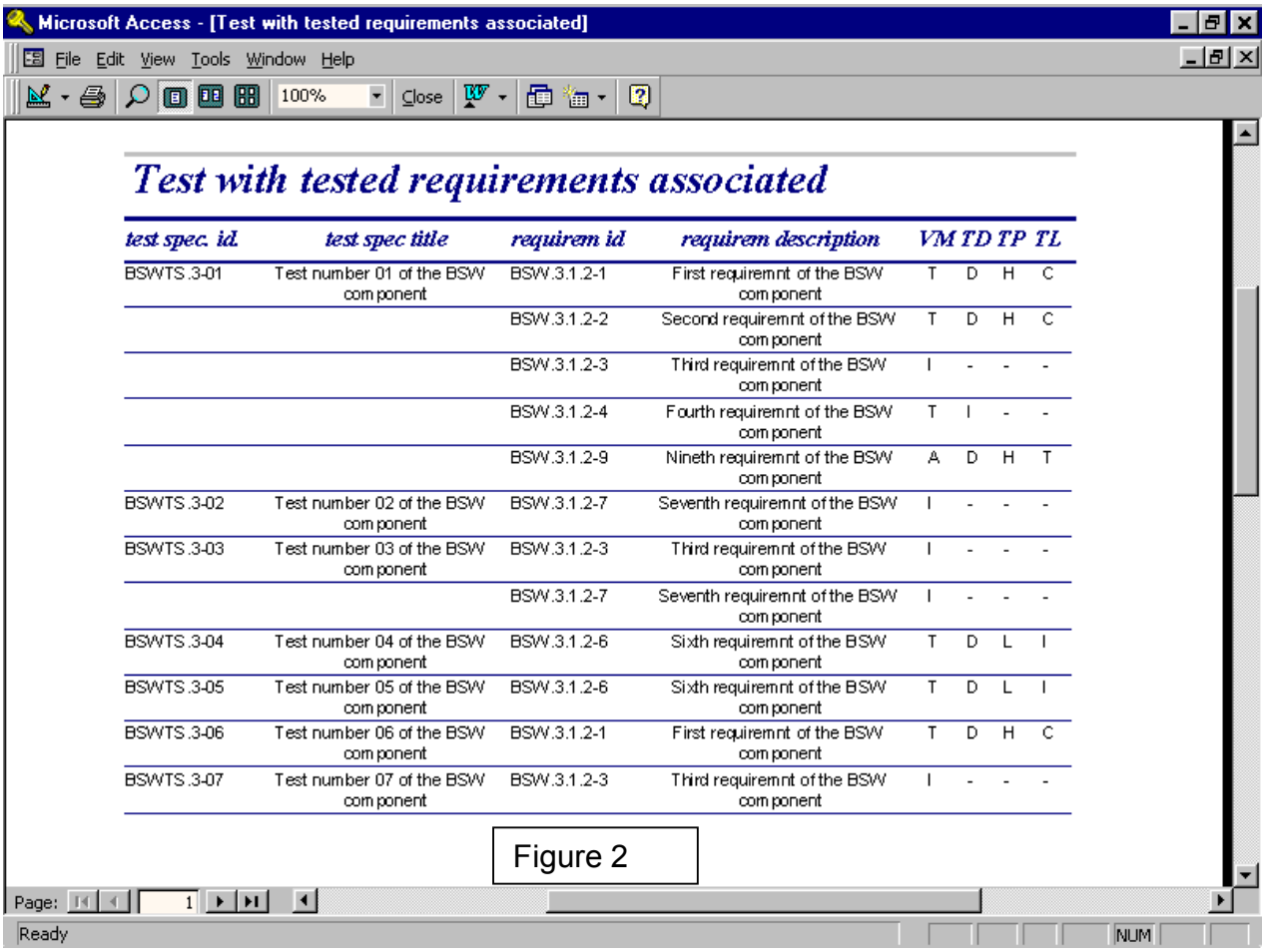
2 Concepts and Definition

Requirement: is a statement that describes a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. It should be written in plain words and good style. A requirement should not be ambiguous. One requirement shall define a single condition or capability at a time. A requirement shall be univocally identified by means of a label.
Architectural Object: an object is a cohesive entity that make up a system that has attributes, behaviour, and possibly a state; it is the result of an architectural design process.
Forward traceability: allows to relate each single requirement toward the following development and validation phases. It helps in showing requirements completeness and duplication, if any.
Backward traceability: it can be logically considered as the inverse operation of the forward. Outputs that cannot be traced to inputs may be superfluous, unless it is acknowledged that the input were incomplete.
Both kinds of tracing are often required by main software quality assurance standards (see [1], [2] , [3] , [4]).
Many-to-many relationship: a relationship between two tables is of many-to-many type (M:N in fig. 1) when one record in either table can relate to many records in the other table.
Requirements definition process: the process of requirements definition starts at system level and proceeds toward lower levels, i.e. subsystem, software, architectural design.
Similarly the relationship attribution process relates:
- subsystem requirements to system,
- software requirements to subsystem requirements,
- architectural design objects to software requirements,
- test specification to related requirements (at the same level).

Each requirement in a document at a level below the system, must refer to one or more requirements specified in one or more documents at the immediately upper level.
Trace Matrix: Tracing is normally shown by matrix listing the correspondence between two entities belonging to different levels; e.g.: list test specifications in the leftmost column of a table and their corresponding requirements specifications in the rightmost column (see fig. 2).
Missing entries in the matrix may display incompleteness.
Cross trace matrix: Cross trace matrix is the one that implies at least two levels of relationship (e.g.: printing of a trace matrix showing the relationship among software tests and subsystem or system requirements using the intermediate relationship with software requirements – see fig. 1 for relationships).

3 Methodology description

The methodology includes a set of practices based on appropriate documents formatting:



test spec. id.	test spec title	requirem id	requirem description	VM	TD	TP	TL
BSWTS.3-01	Test number 01 of the BSW component	BSW.3.1.2-1	First requiremnt of the BSW component	T	D	H	C
		BSW.3.1.2-2	Second requiremnt of the BSW component	T	D	H	C
		BSW.3.1.2-3	Third requiremnt of the BSW component	I	-	-	-
		BSW.3.1.2-4	Fourth requiremnt of the BSW component	T	I	-	-
		BSW.3.1.2-9	Nineth requiremnt of the BSW component	A	D	H	T
BSWTS.3-02	Test number 02 of the BSW component	BSW.3.1.2-7	Seventh requiremnt of the BSW component	I	-	-	-
BSWTS.3-03	Test number 03 of the BSW component	BSW.3.1.2-3	Third requiremnt of the BSW component	I	-	-	-
		BSW.3.1.2-7	Seventh requiremnt of the BSW component	I	-	-	-
BSWTS.3-04	Test number 04 of the BSW component	BSW.3.1.2-6	Sixth requiremnt of the BSW component	T	D	L	I
BSWTS.3-05	Test number 05 of the BSW component	BSW.3.1.2-6	Sixth requiremnt of the BSW component	T	D	L	I
BSWTS.3-06	Test number 06 of the BSW component	BSW.3.1.2-1	First requiremnt of the BSW component	T	D	H	C
BSWTS.3-07	Test number 07 of the BSW component	BSW.3.1.2-3	Third requiremnt of the BSW component	I	-	-	-

Figure 2

- automatic loading of requirements, architectural objects and test specification in database tables;
- automatic setting of relationships among requirements at different levels;
- automatic setting of relationships among test specifications and related requirements;
- automatic setting of relationships among architectural design objects and software requirements.

Cross trace matrix: Cross trace matrix is the one that implies at least two levels of relationship (e.g.: printing of a trace matrix showing the relationship among software tests and subsystem or system requirements using the intermediate relationship with software requirements – see fig. 1 for relationships).
It is very important to highlight that the approach described in the paper allowed a great saving in time and a good reliability in database information management.

After the database has been populated a set of Man Machine Interfaces allows the interactive relationship management and the production of the trace matrixes and cross trace matrixes.
A set of reports allows trace matrix printing for documentation and check purposes.

3.1. Automatic Items Loading

This section explains the technique adopted to allow automatic population of the database tables containing items to be traced. The description refers to a Software Requirements Document (SRD).
All the requirements written in the document are formed by two parts: a *requirement header* and a *requirement body*. Example:

##BSW.3.1.2-5 Time Tagged Tele Commands shall be executed within 1 ms [T] *<body of the requirement>*

In the SRD each requirement is written using the following syntax pattern:
##XXX.<par.n.>-<prg.><tab><req.h><tab> <v.m.>
where:
XXX is a three letter code indicating the CI,
par.n.: is the paragraph number in the document,
prg.: an unique progressive number within the paragraph,
req.h.: is the requirement header,
v.m. is the validation method.
The validation method is often equal to test and therefore the two words have been used as synonymous.
In the example the fifth requirement of the paragraph 3.1.2 of the software component implementing the Basic Software (BSW) functions is validated by test.
The characters "##" of the *requirement header* were used to allow the automatic recognition, by a procedure, in order to extract and load the complete requirement header row. The body is used, in the document, to give details on requirement and it is not loaded in the database table.
A similar approach has been adopted in the:

- system requirements document using a string such as:
  ##**SYS.3.7.8-5** System Requirement number 5 in the paragraph 3.7.8;
- subsystem requirement document using a string such as:
  ##SSA.6.3.4-7 Requirement number 7 in the paragraph 6.3.4 of the Sub System A;
- test specification document:
  ##SSATS.3-14 test specification 14 in the paragraph 3 related with the Sub System A.

The string between the "##" characters and the header, bold faced, acts as an unique label (i.e. it is a primary key), and is arbitrary, i.e. the database behaviour is not based on the label key string structure. There is a unique limitation on the label length that is maximum 30 characters.
The end of the label is determined by a <tab> character.
During loading a check for the uniqueness of the label key is made, i.e. double key are refused and recorded in a log-file as an error.
It is recommended to use different "letter codes" in the key for different documents whose items are loaded in the same table; this will avoid key duplication during items loading from different documents.
The approach described allowed to load database tables automatically with system requirements, subsystem requirements, software requirements, architectural objects and with test specifications.
The number of the documents allowed for each table is arbitrary (see logical database schema in figure 1).

The maximum number of items (requirements, test specifications and architectural design objects) is limited by the maximum number of records for each table, that is well above the needs. Up to five thousand requirements, and two thousand test specifications have been managed in the most important application till today.

3.2. Automatic Relationships Loading

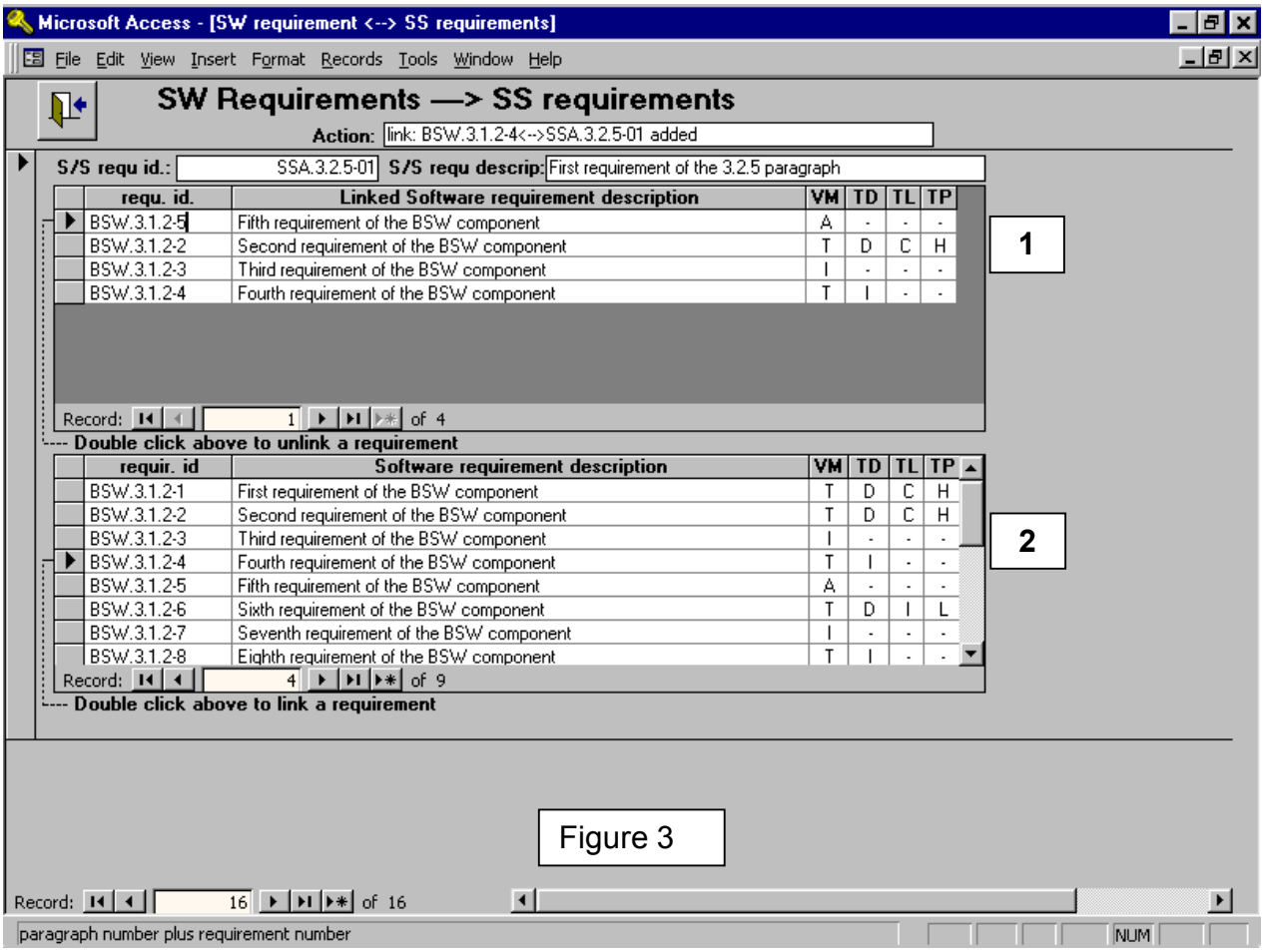
The relationship between items at different levels must be loaded in the database in order to allow the correspondence management, check and report (see fig. 1 for relationship schema).

Following the rules foreseen by the methodology the loading can be accomplished automatically. A detailed description follows:
##SSA.4.1.2-10 <Requirement number 10 of the paragraph 4.1.2 of the subsystem A >
#$SYS.3.5.4-3
#$SYS.3.7.3-7
The example must be read in the following way. The subsystem requirement 4.1.2-10 belonging to the "Subsystem A" has been derived from the requirements at system level SYS.3.5.4-3 and SYS.3.7.3-7.
With this approach the database loading procedure at first recognise and loads the requirement SSA.4.1.2-10 and then its links with the system requirements loading the links in the appropriate internal table.

In order to better clarify the approach another example follows:
##SSATS.10-12 <SSA test 10-12 specification header>
#$SSA.3.4.1.5-7
#$SSA.2.10.5.2-3
The database load routine recognises the test specification 10-12 and loads its header in the subsystem test table therefore the relationships with the two subsystem requirements SSA.3.4.1.5-7 and SSA.2.10.5.2-3 are loaded. As consequence the database has loaded the information that the requirements SSA.3.4.1.5-7 and SSA.2.10.5.2-3 are tested by the test SSATS.10-12 and vice versa.
The same approach has been adopted in all the other relationships drawn in figure 1.
No checks of relationship key existence is made during loading operations. Consistency checks can be accomplished by adequate reports.

3.3. Interactive relationship management

In the figure 3 the MMI designed to manage the relationships between subsystem "A" (SSA) requirements and software requirements is drawn.
The form allows a quick mean to manage relationships.
A brief explanation (starting from the bottom of the figure 3) is given in the forthcoming rows.
The MMI contains two sub-forms. The sub-form 2 lists all the software requirements belonging to the Subsystem "A". The sub-form 1 lists all the software requirements that has been related with the subsystem A requirement specification SSA.3.2.5-01.
The current subsystem A requirement is shown in the row of the form labelled with “S/S requ id” followed by the requirement description in the same row.
The row “Action” shows the last operation performed on a link.
Two operations are available:



Delete a link: to indicate that a relationship has to be cancelled.
The Add/Delete operations were managed by “mouse double click”, on the appropriate MMI sub-form, in a very easy and rapid way. After each operation a refresh of the sub-form 1 is executed, i.e. the one indicating linked BSW requirements. In the figure 3 the Add operation between the S/S requirement 'SSA.3.2.5-01' is shown in the window "S/S requ. id" and the software requirement BSW.3.1.2-4 double clicked in the sub-form 2 is shown. Note that the row labelled with BSW.3.1.2-4 has been added in the sub-form 1.
Finally the figure 3 meaning is that the four requirements of BSW CI (3.1.2-2, -3, -4 and –5) have been derived from the S/S requirement SSA.3.2.5-01.
In a very similar way it is possible, for example, to delete the requirement relationship:
BSW.3.1.2-5<->SSA.3.2.5-01
to accomplish this action it is necessary to double click on the requirement BSW 3.1.2-5 in the sub-form 1.

4 Database Internal Structure

The relationship implemented in the database is of many-to-many type (M:N). For example a requirement can be related with one or more test specifications and a test specification refers one or more requirements.
Similarly for relationship among requirements: a subsystem requirement can be related with one or more system requirement and vice versa. In the figure 4 a representation of the implemented

M:N relationship among tables is sketched (fig. 4 doesn't contain all tables of documents in fig. 1 for lack of room). The management of the N:M relationship has been solved loading in a third hidden table (see fig. 4), for each relationship, the primary keys of the related entities.

The operation “Add a link” is managed inserting a row, containing the primary keys of related items, in the appropriate table.

The operation “Delete a link” is managed deleting a row, containing the primary keys of related items, from the table.

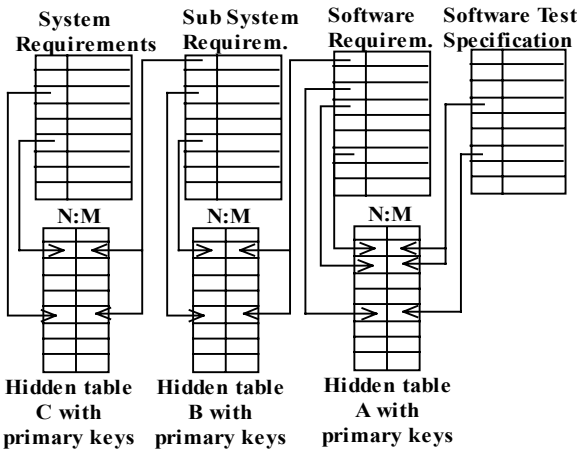


Fig. 4 - Tables layout structure

The insertion/deletion of the records containing the relationship keys (tables A, B and C) is hidden to the user and completely controlled by the application using the MMI interface. The automatic management assures the consistency of the process.

5 Requirements attributes

The in-house development allowed to customise attributes given to the traced item. As an example the attributes added with software requirements are described (see fig 2 and 3):

VM: Validation Method. Possible values: A = Analysis, I = Inspection, T = Test;

TD: Test Design. Applicable only if VM=T. Can assume the values:

D = Direct - at least one test exists to validate,

I = Indirect - the requirement is linkable to another test specification;

TP: Test Priority. Applicable only if VM=T. Can assume the values:

H = High - the test procedure, execution and verification has to be implemented first,

L = Low - the test procedure implementation execution and verification can be postponed,

TL: Test Level. Applicable only if VM=T. Can assume the values:

C = Component: the related test will be executed at component level only,

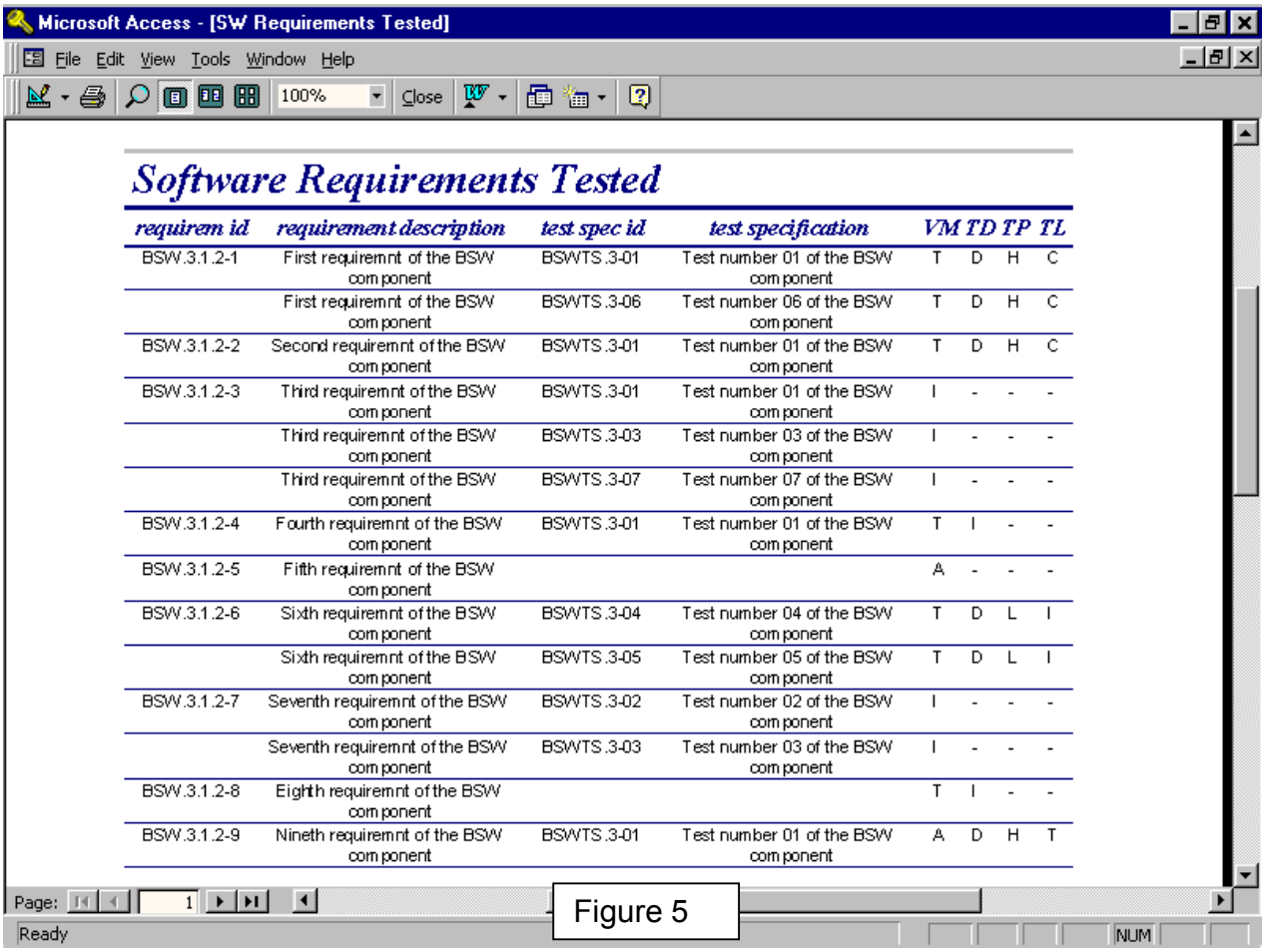
I = Integrated: the related test will be executed on the integrated test platform, with the real hardware,

T = Total: the test will be conducted at both levels C and I.

6 Reports

The database managing the relationships among requirements, architectural objects and test specifications allowed to produce the following reports:

- listing of all software requirements with related architectural design objects for each CI;
- listing of all requirements at system, subsystem and software level with related test specifications; see as example software requirements with related test in figure 5. Note that the requirements BSW.3.1.2-5 and 3.1.2-8 do not have a test specification identified on their row; this implies that the two requirements have not a test assigned. The same result could be obtained with a check report that lists only requirements that do not have test specifications assigned;
- listing of all requirements not related to any architectural design object (should be empty); any existing unlinked requirement should be corrected or justified;



Software Requirements Tested

requirem id	requirement description	test spec id	test specification	VM	TD	TP	TL
BSW.3.1.2-1	First requiremnt of the BSW component	BSWTS.3-01	Test number 01 of the BSW component	T	D	H	C
	First requiremnt of the BSW component	BSWTS.3-06	Test number 06 of the BSW component	T	D	H	C
BSW.3.1.2-2	Second requiremnt of the BSW component	BSWTS.3-01	Test number 01 of the BSW component	T	D	H	C
BSW.3.1.2-3	Third requiremnt of the BSW component	BSWTS.3-01	Test number 01 of the BSW component	I	-	-	-
	Third requiremnt of the BSW component	BSWTS.3-03	Test number 03 of the BSW component	I	-	-	-
	Third requiremnt of the BSW component	BSWTS.3-07	Test number 07 of the BSW component	I	-	-	-
BSW.3.1.2-4	Fourth requiremnt of the BSW component	BSWTS.3-01	Test number 01 of the BSW component	T	I	-	-
BSW.3.1.2-5	Fifth requiremnt of the BSW component			A	-	-	-
BSW.3.1.2-6	Sixth requiremnt of the BSW component	BSWTS.3-04	Test number 04 of the BSW component	T	D	L	I
	Sixth requiremnt of the BSW component	BSWTS.3-05	Test number 05 of the BSW component	T	D	L	I
BSW.3.1.2-7	Seventh requiremnt of the BSW component	BSWTS.3-02	Test number 02 of the BSW component	I	-	-	-
	Seventh requiremnt of the BSW component	BSWTS.3-03	Test number 03 of the BSW component	I	-	-	-
BSW.3.1.2-8	Eighth requiremnt of the BSW component			T	I	-	-
BSW.3.1.2-9	Nineth requiremnt of the BSW component	BSWTS.3-01	Test number 01 of the BSW component	A	D	H	T

Figure 5

- listing of all requirements not related with at least one test specification (should be empty); any existing unlinked requirement should be corrected or justified;

All the reports have been used and checked for consistency and correctness.

7 Code amount and other data

The total amount of statements has been evaluated in more than 3500 and 25 SQL queries.
The following elements have been designed: 16 "Tables", 40 "Forms",30 "Reports".
The effort to develop the application can be estimated in seven hundred hours.

8 Future developments

Possible enhancements that are foreseen are the following:

- Control of test evolution in terms of test design, run, result validation;
- Support to problem reporting raised during tests run and the relevant evolution;
- Requirement stability control, to trace evolution during the project lifecycle;
- Support multiple formats for input documents during database population.

Currently the release 5.0 is under design. It has been completely redesigned in order to be fully parametric, i.e. it will be able to manage many-to-many relationships between sets of documents not necessarily related with requirements or tests specifications.

Acronyms

BSW = Basic SoftWare
CI = Configuration Item
ESA = European Space Agency
MMI = Man Machine Interface
M:N = Many-to-many
RTM = Requirements Trace Methodology
SRD = Software Requirements Document
SSATS= Test Specification of S/S A
S/S = Sub System
SYS = System
VM = Validation Method

Bibliography

[1] MIL-STD-498 Software Development and Documentation
5 December 1994
[2] MIL-STD-498 Guidebook-Application and Reference
31-January 1996
[3] ESA PSS-05-0 Software Engineering Standards Issue 2
February 1991
[4] ECSS-Q-80-A Space Product Assurance
13 April 1999

Requirement Trace Application

Francesco Piazza

Introduction

- This presentation is aimed at describing a facility designed to trace requirements through a development life cycle, i.e.
  - System requirements to S/S requirements
  - S/S requirements to software requirements
  - S/S requirements to S/S test procedure
  - Software requirements to S/W test procedure
  - Software requirements to architectural design

The application is based on the database Access '97 by Microsoft

- It has been designed in order to fulfil a mandatory requirement included in many software standards such as
  - PSS and ECSS series by ESA
  - MIL-STD-498 military standard

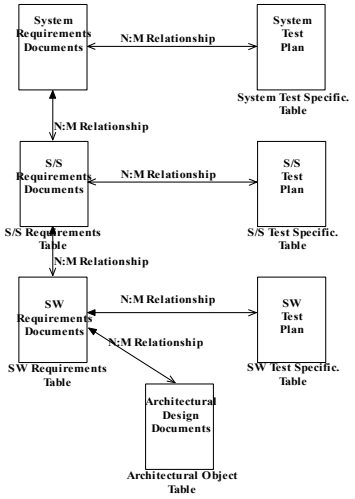
Alenia SPAZIO
A Finmeccanica Company

Documents relationships related with database schema

- The figure on the right shows the document structure that can be managed with the application
- The lines show the relationships (N:M type) between the documents that the system can trace
- The figure shows that tracing is possible among:

...design phases from system below

...test phases against requirement



Syntax and structure of documents (I)

- The slide shows below an example of the syntax used to write items (software requirement) and relevant relationship with s/w test spec
- The item row is followed by the related test specification and by the body of the items itself
- It is possible to specify more than one parent
- The characters ## and #$ are used to allow database automatic load

Example:

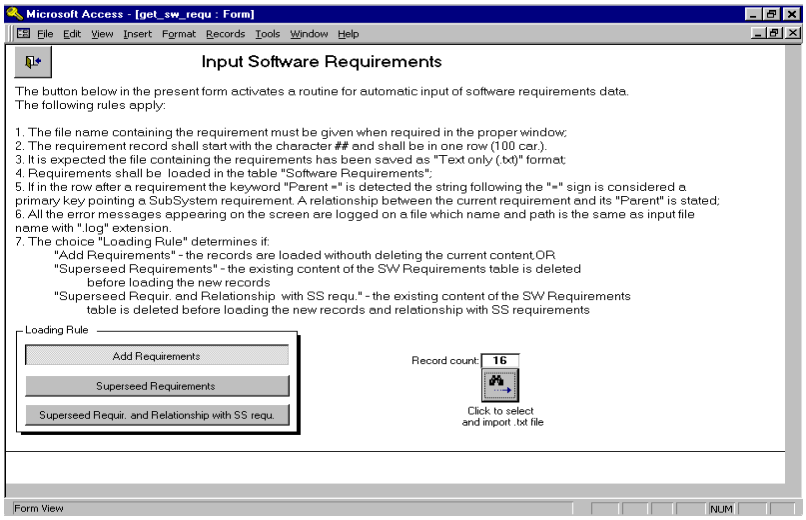
- *##BSW.3.1.2-5 Time Tagged TCs shall be executed within 1 ms [T]*
- #$TSSW.4-7, TSSW.5-1
- *<body of the software requirement>*

Syntax and structure of documents (II)

- The slide shows below an example of the syntax used to write items (software requirement) and relevant relationship with S/S requir. spec
- The requirement row is followed by the parent specification and by the body of the requirement itself
- It is possible to specify more than one parent
- The characters ## and #$ are used to allow database automatic load

- Example:
- *##BSW.3.1.2-5 Time Tagged TCs shall be executed within 1 ms [T]*
- #$SSA.5.1.2-4, SSA.3.4.5.6-1
- *<body of the the software requirement>*

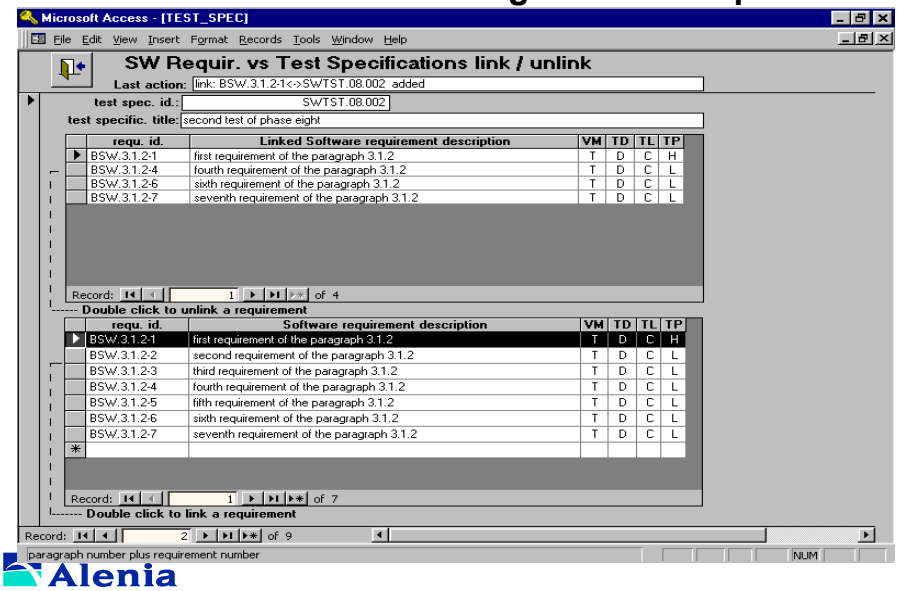
MMI to load software requirements



Loading items and relationships

- The previous slide shows the MMI used to load items (software requirements in this case) and relationships in the database
- Requirement can be loaded in addition to already existing items or supersede existing items and/or relationships
- Controls are accomplished on:
  - duplicate identifiers (that are forbidden)
  - duplicate relationships
  - syntax errors on item specification
- No controls on relational integrity are made at loading time
- All the errors are logged on file to allow error collection
- Controls on not linked requirements or test can be accomplished through proper check reports

MMI interface to manage relationships

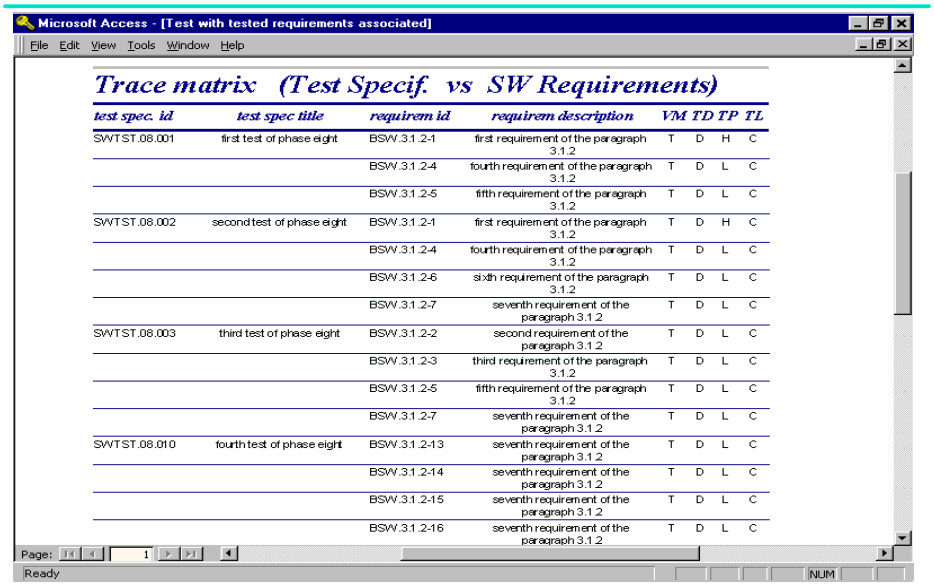


MMI interface to explanation

- The previous slide shows the MMI interface used to manage relationships between software requirements and related test specification
- The figure shows on the top the s/w test identifier and test description whose relationships will be managed
- The frame immediately below shows the already related requirements (linked requirements)
- The frame at the bottom shows all the software requirements that can be related with the current test
- Double clicking on the frame above unlinks a requirement with the test specification. The frame is immediately updated
- Double clicking on the frame below links a requirement with the test specification shown

Reports

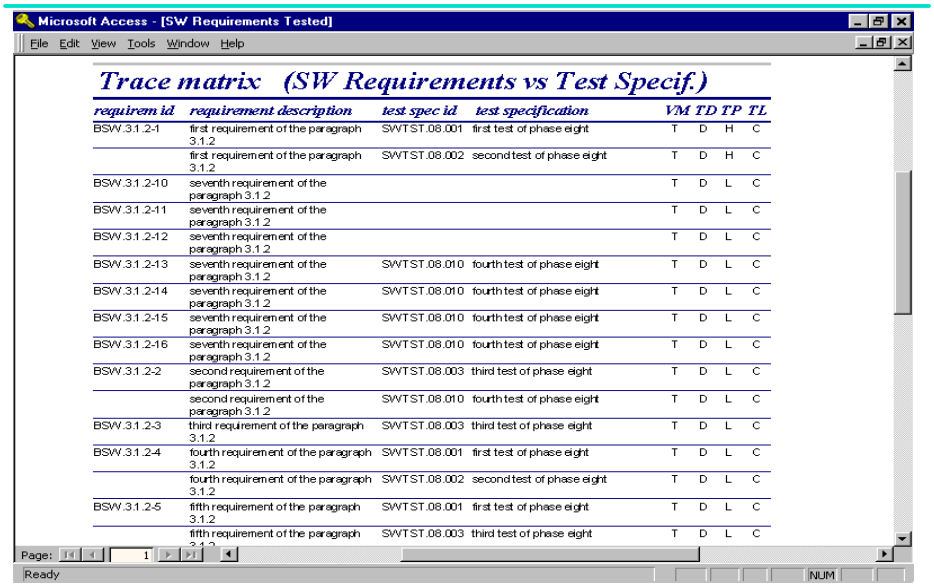
- Next slides show three reports:
  - a Trace Matrix from SW test specification vs SW requirements. The requirements tied to each test can be used as input specification for the test design and for the test result evaluation
  - a Trace Matrix from SW requirements vs SW tests. This report show requirement not related to a test (if any) and, for each requirement, shows the amount of test effort
  - a Cross Trace Matrix from SW requirements vs S/S tests through S/S requirements.
    Cross Trace Matrix is a report based on a query that involves two level of relationship among three tables. The report shows how each sw requirement is involved in S/S testing effort
- Other report are available such as:
  - list of SW requirement, S/S requirement, SW tests, S/S tests
  - check reports showing SW or S/S requirements not tested
  - check reports showing SW or S/S requirements not flown down to design



Trace matrix (Test Specif. vs SW Requirements)

test spec. id	test spec title	requirem id	requirem description	VM	TD	TP	TL
SWTST.08.001	first test of phase eight	BSW.3.1.2-1	first requirement of the paragraph 3.1.2	T	D	H	C
		BSW.3.1.2-4	fourth requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-5	fifth requirement of the paragraph 3.1.2	T	D	L	C
SWTST.08.002	second test of phase eight	BSW.3.1.2-1	first requirement of the paragraph 3.1.2	T	D	H	C
		BSW.3.1.2-4	fourth requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-6	sixth requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-7	seventh requirement of the paragraph 3.1.2	T	D	L	C
SWTST.08.003	third test of phase eight	BSW.3.1.2-2	second requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-3	third requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-5	fifth requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-7	seventh requirement of the paragraph 3.1.2	T	D	L	C
SWTST.08.010	fourth test of phase eight	BSW.3.1.2-13	seventh requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-14	seventh requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-15	seventh requirement of the paragraph 3.1.2	T	D	L	C
		BSW.3.1.2-16	seventh requirement of the paragraph 3.1.2	T	D	L	C

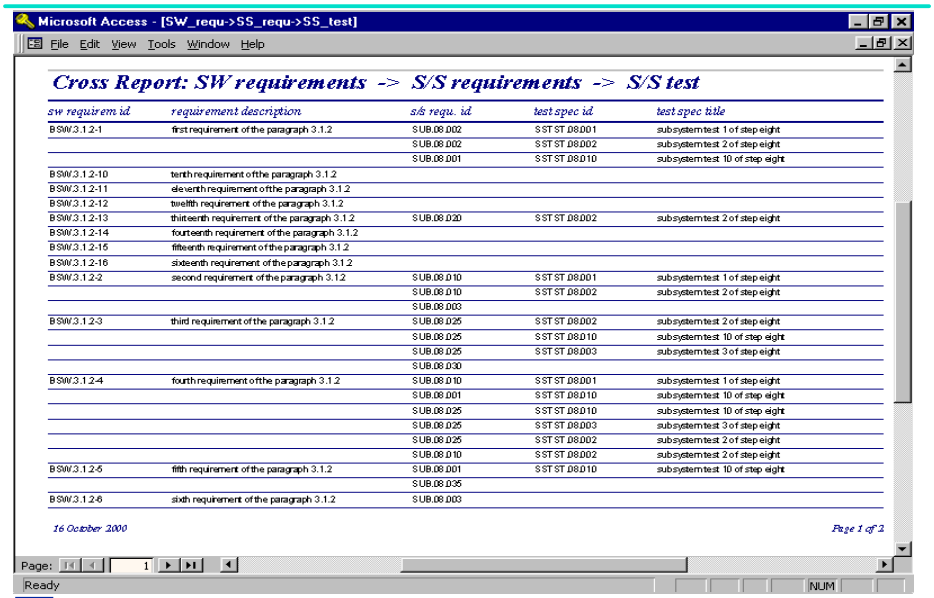
Alenia
SPAZIO
A Finmeccanica Company



Trace matrix (SW Requirements vs Test Specif.)

requirem id	requirement description	test spec id	test specification	VM	TD	TP	TL
BSW.3.1.2-1	first requirement of the paragraph 3.1.2	SWTST.08.001	first test of phase eight	T	D	H	C
	first requirement of the paragraph 3.1.2	SWTST.08.002	second test of phase eight	T	D	H	C
BSW.3.1.2-10	seventh requirement of the paragraph 3.1.2			T	D	L	C
BSW.3.1.2-11	seventh requirement of the paragraph 3.1.2			T	D	L	C
BSW.3.1.2-12	seventh requirement of the paragraph 3.1.2			T	D	L	C
BSW.3.1.2-13	seventh requirement of the paragraph 3.1.2	SWTST.08.010	fourth test of phase eight	T	D	L	C
BSW.3.1.2-14	seventh requirement of the paragraph 3.1.2	SWTST.08.010	fourth test of phase eight	T	D	L	C
BSW.3.1.2-15	seventh requirement of the paragraph 3.1.2	SWTST.08.010	fourth test of phase eight	T	D	L	C
BSW.3.1.2-16	seventh requirement of the paragraph 3.1.2	SWTST.08.010	fourth test of phase eight	T	D	L	C
BSW.3.1.2-2	second requirement of the paragraph 3.1.2	SWTST.08.003	third test of phase eight	T	D	L	C
	second requirement of the paragraph 3.1.2	SWTST.08.010	fourth test of phase eight	T	D	L	C
BSW.3.1.2-3	third requirement of the paragraph 3.1.2	SWTST.08.003	third test of phase eight	T	D	L	C
BSW.3.1.2-4	fourth requirement of the paragraph 3.1.2	SWTST.08.001	first test of phase eight	T	D	L	C
	fourth requirement of the paragraph 3.1.2	SWTST.08.002	second test of phase eight	T	D	L	C
BSW.3.1.2-5	fifth requirement of the paragraph 3.1.2	SWTST.08.001	first test of phase eight	T	D	L	C
	fifth requirement of the paragraph 3.1.2	SWTST.08.003	third test of phase eight	T	D	L	C

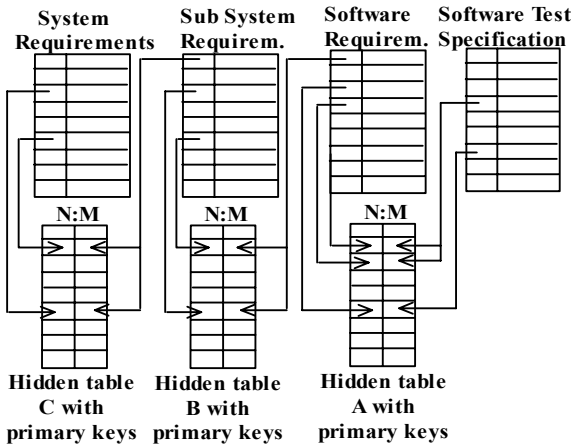
Alenia
SPAZIO
A Finmeccanica Company



Alenia
SPAZIO

Database tables layout structure

The figure on the right shows tables containing data to be traced and the hidden tables that are managed by the application for tracing purposes



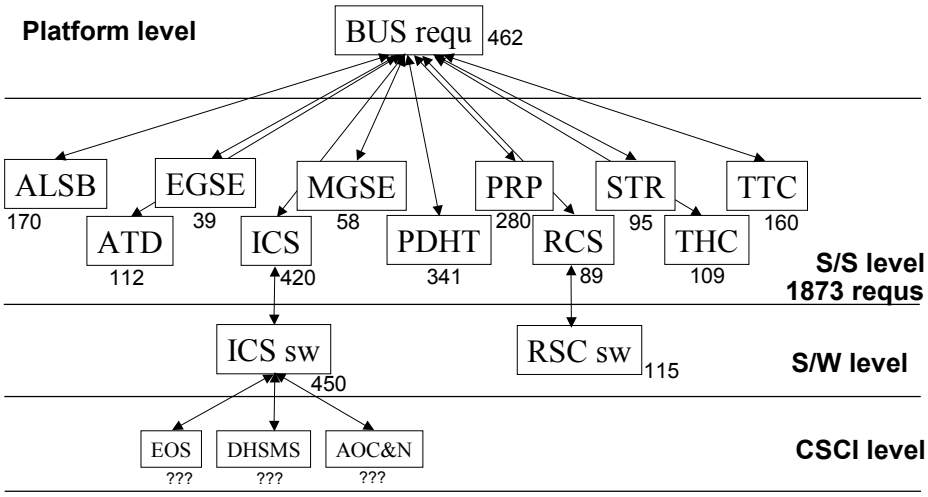
Alenia
SPAZIO

Requirement Trace Application Evolution

- A new release completely redesigned has been issued at the end of June
- The new release has been speeded up by a new project with a structure more complex than the previous
- It is based on the observation that a tool able to manage N:M relation in a completely parametric way would be satisfactory for trace management purposes
- The tool takes as input from a configuration table:
  - **the names of the two tables involved in the relationship and of the link table**
  - **the names of the forms and of the reports to be used**
  - **the names of the headers showing the name of each table**
- The new approach is able to manage an unlimited number of relationships; only one relationship is managed at a time
- next slide shows the structure of a system currently traced

Alenia SPAZIO

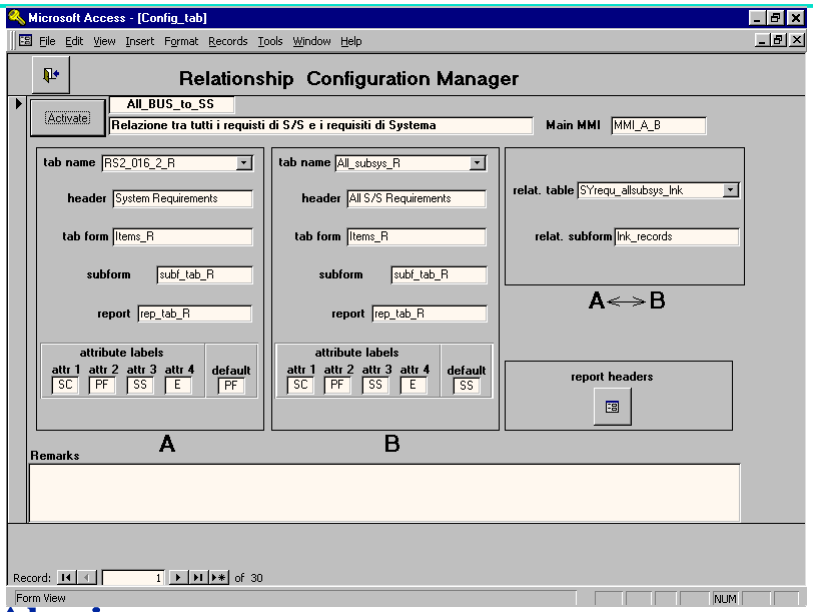
Bus vs S/S Requirements

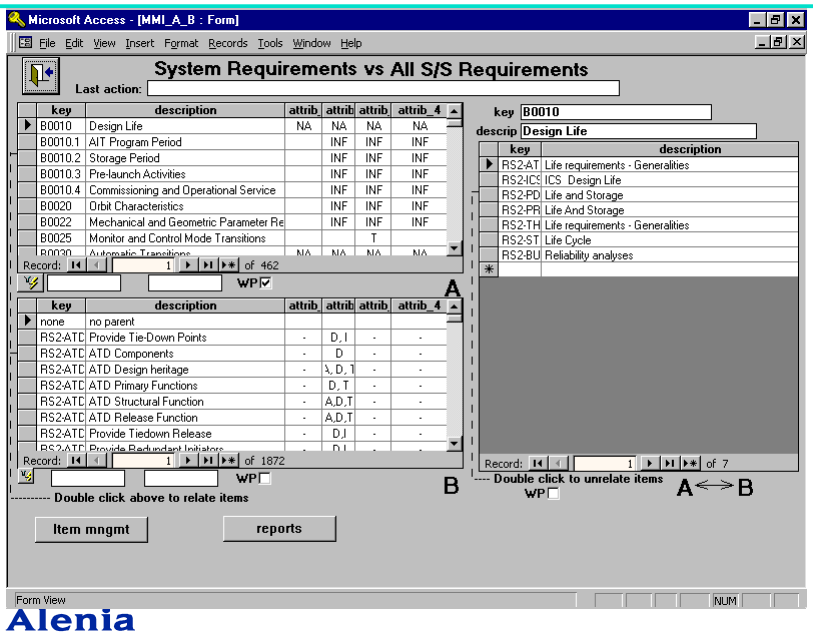


Alenia SPAZIO

Relationship Configuration

- Next slide shows the MMI used to fill in the table used to manage relationships
- The table considered are always generically named as table A and table B; the name of a third table to memorise link relationship information is shown
- It is possible to see that parametric values are on:
  - data table names and link table name,
  - sub-forms used to interactively manage information
  - reports used to show relationships
- It is also possible to manage some attributes, tied with each record, that in the current application are dedicated to specify the test level that currently are:
  - SC = spacecraft PF = Platform SS = subsystem
  - E = Equipment SW = software





Alenia SPAZIO

A Finmeccanica Company

Item management

- Previous slide shows the MMI used to link/unlink items present in table A and B (see the two sub-forms on the left);
  linked items are shown on the subform on the rigth
- At the bottom of the MMI two buttons are shown:
  - item management: allow the activation of the record input, deletion and change
  - reports: allow the activation of the available reports, Trace Matrix, Check Matrix and the generation of word .rft format file to export data

Conclusion

- The approach adopted in the current release of the application will allow, with a little effort, the management of other items, but this area is still under development

Alenia SPAZIO

A Finmeccanica Company

QWE2000 Session 6A

Jacobus DuPreez & Lee D. Smith
(ARM Ltd)

SPI: A Real-World Experience

Key Points

- The rationale behind SPI
- A success story in the making
- Essential attributes of an effective SPI model

Presentation Abstract

After exposure to all phases and various models of large-scale software engineering, the presenter became increasingly fascinated by the challenge of predictably engineering quality in software, and his search for a climate in which genuine SPI would "grow" well led him to join the software group of ARM Ltd (http://www.arm.com) early 1999 as a full-time internal SPI consultant. The paper relates his successes in this team since then and highlights a number of key prerequisites to viable SPI.

The concepts discussed would be of interest to anyone involved in SPI and/or related initiatives in growing software teams. The presenter's experience prior to joining ARM includes the setting up from scratch and management of a test department in a medium-sized software house.

About the Speaker

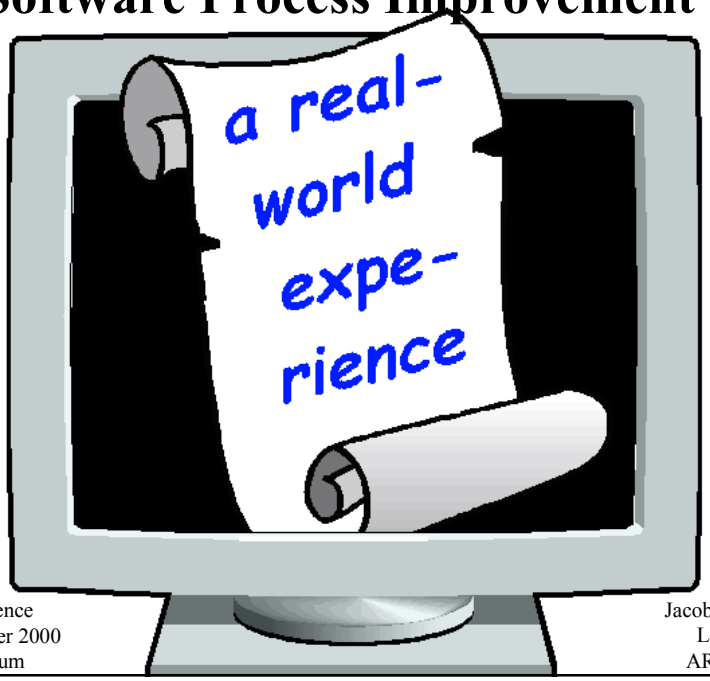
Jacobus C. Du Preez, BA Hons (French)

In 1989, following a career in translating/interpreting, Du Preez successfully produced an automated translation prototype, which led to a new career in software development. He first documented a 4GL, then developed and presented a training package for it, then worked as an applications developer, and ended up in senior support and test roles. This hands-on experience in all development life cycle phases left him fascinated by the challenge of predictably engineering genuine quality in software, in spite of the particular dynamics of the industry. He joined ARM in March 1999, where his role currently is to focus exclusively on software process improvement.

Lee D. Smith, MA (Cantab) (Mathematics, BA 1974)

Smith has been involved in the construction of large-scale software systems for more than 25 years. In 1978 he became an academic Computer Scientist (Edinburgh University) and in 1983 he moved to Acorn Computers to manage a VLSI design tools project. He moved into the field of compilers and development tools at the end of 1987, where he works until today. Smith has been with ARM since its start-up days, and he has been been involved in its software process improvement initiatives for more than five years. Smith is also co-author of "Challenges in Cross-development", an article published in the Jul/Aug 1997 IEEE Micro (http://computer.org/micro/mi1997/m4toc.htm).

Software Process Improvement



4 th Annual
International
Quality Week
Europe Conference
22-24 November 2000
Brussels, Belgium

Jacobus DuPreez
Lee D. Smith
ARM Ltd, UK

INTRODUCTION

INTRODUCTION

- THE RATIONALE BEHIND SPI

INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING

INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING
- ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING
- ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

CONCLUSION

INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING
- ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

CONCLUSION

INTRODUCTION

- [x] introduction to the company

INTRODUCTION

- introduction to the company
- introduction to the context

INTRODUCTION

- introduction to the company
- introduction to the context
- introduction to the story

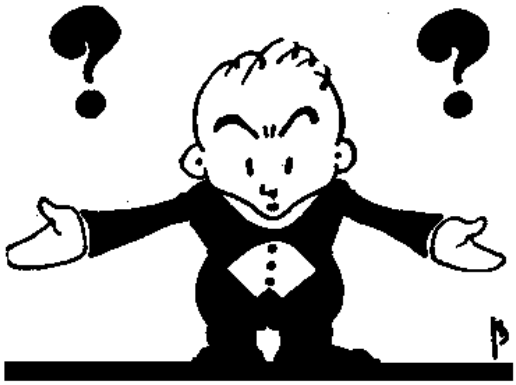
INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING
- ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

CONCLUSION

THE RATIONALE BEHIND SPI

- loops within loops

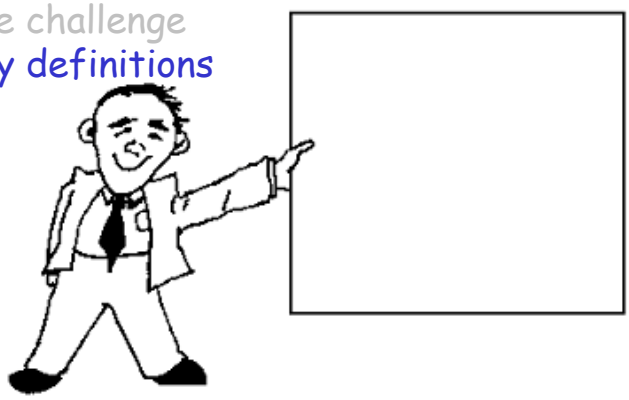


THE RATIONALE BEHIND SPI

- loops within loops
- the challenge

THE RATIONALE BEHIND SPI

- loops within loops
- the challenge
- key definitions

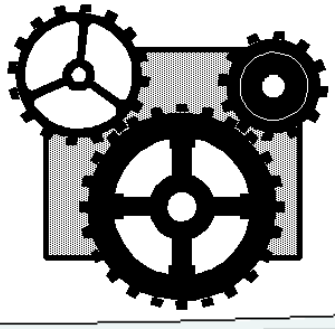


THE RATIONALE BEHIND SPI

- loops within loops
- the challenge
- key definitions
- engineering vs process

THE RATIONALE BEHIND SPI

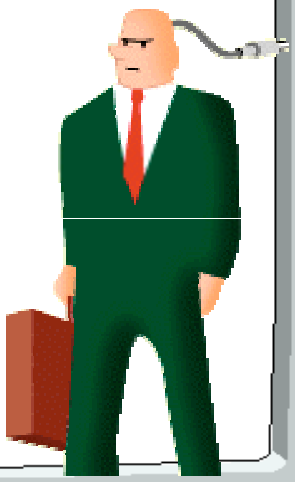
- loops within loops
- the challenge
- key definitions
- engineering vs process
- software process goodness



THE RATIONALE BEHIND SPI

- loops within loops
- the challenge
- key definitions
- engineering vs process
- software process goodness

☑ software process reuse



THE RATIONALE BEHIND SPI

- loops within loops
- the challenge
- key definitions
- engineering vs process
- software process goodness
- software process reuse

☑ measuring SPI

THE RATIONALE BEHIND SPI

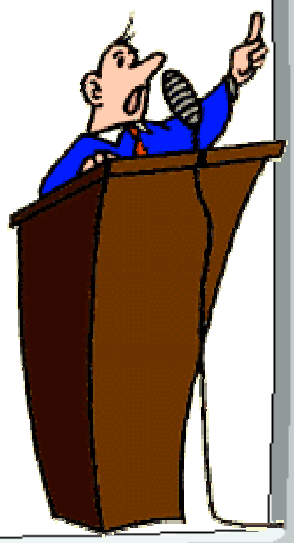
- loops within loops
- the challenge
- key definitions
- engineering vs process
- software process goodness
- software process reuse
- measuring SPI
- the essential nature of SPI

THE RATIONALE BEHIND SPI

- loops within loops
- the challenge
- key definitions
- engineering vs process
- software process goodness
- software process reuse
- measuring SPI
- the essential nature of SPI
- two crucial factors

THE RATIONALE BEHIND SPI

- loops within loops
- the challenge
- key definitions
- engineering vs process
- software process goodness
- software process reuse
- measuring SPI
- the essential nature of SPI
- two crucial factors
- the case for SPI



INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING
- ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

CONCLUSION

A SUCCESS STORY IN THE MAKING

- the SW group

A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains

A SUCCESS STORY IN THE MAKING

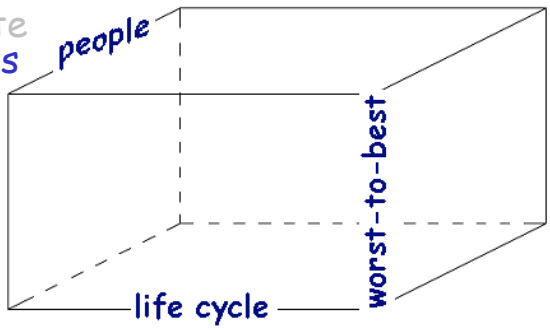
- the SW group
- growth pains
- ☑ full-time SPI

A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- ☑ a suitable climate

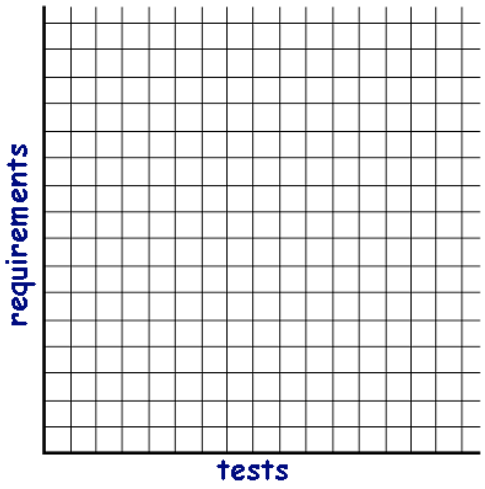
A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions



A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point

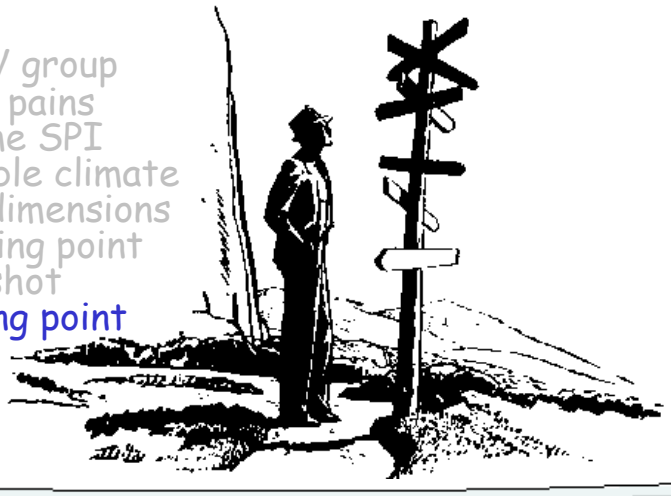


A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot

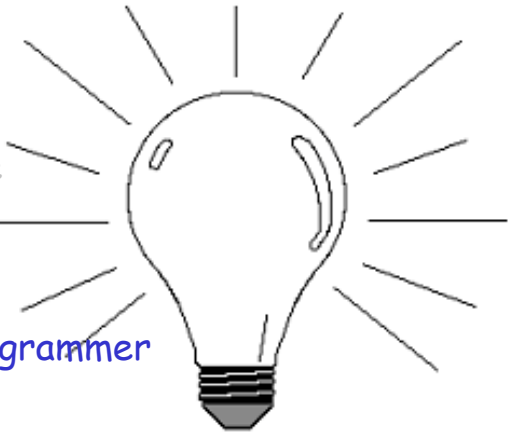
A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point



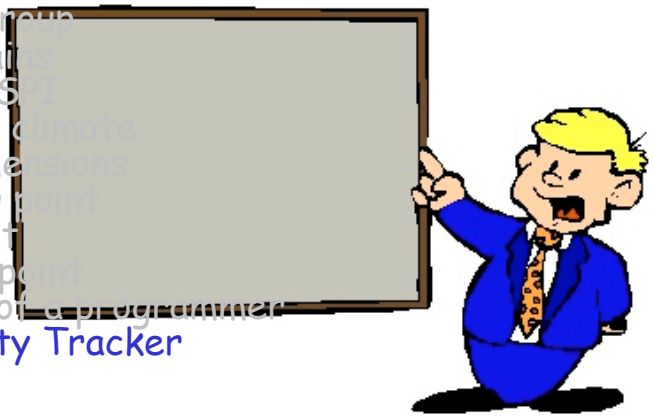
A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point
- the mind of a programmer



A SUCCESS STORY IN THE MAKING

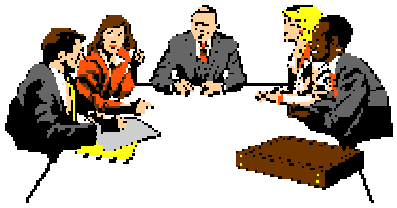
- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point
- the mind of a programmer
- the Quality Tracker



A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point
- the mind of a programmer
- the Quality Tracker

☑ the inspection process



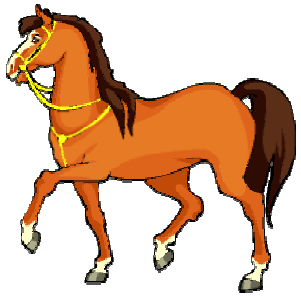
A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point
- the mind of a programmer
- the Quality Tracker
- the inspection process

☑ success breeding success

A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point
- the mind of a programmer
- the Quality Tracker
- the inspection process
- success breeding success
- ☑ requirements management



A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point
- the mind of a programmer
- the Quality Tracker
- the inspection process
- success breeding success
- requirements management
- ☑ a success story

A SUCCESS STORY IN THE MAKING

- the SW group
- growth pains
- full-time SPI
- a suitable climate
- three dimensions
- a starting point
- a long shot
- a turning point
- the mind of a programmer
- the Quality Tracker
- the inspection process
- success breeding success
- requirements management
- a success story
- next steps

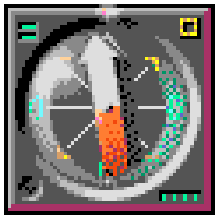
INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING
- ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

CONCLUSION

ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

- orientation



ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

- orientation
- popularity

ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

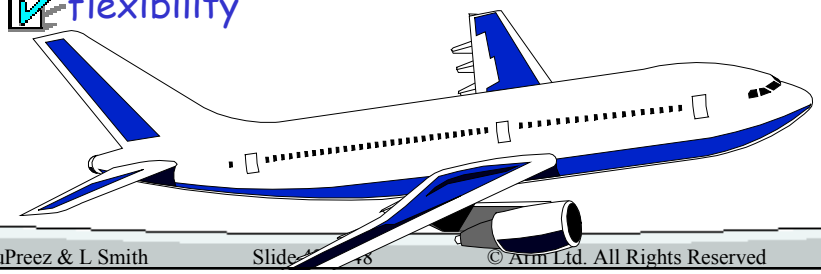
- orientation
- popularity
- clarity

ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

- orientation
- popularity
- clarity
- appropriateness

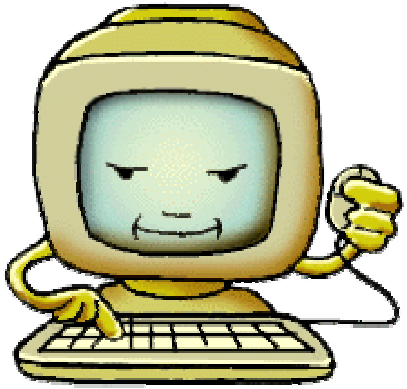
ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

- orientation
- popularity
- clarity
- appropriateness
- flexibility



ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

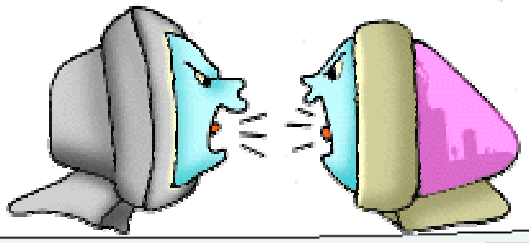
- orientation
- popularity
- clarity
- appropriateness
- flexibility
- automation



- simplicity

ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

- orientation
- popularity
- clarity
- appropriateness
- flexibility
- automation
- simplicity
- ☑ coherence



INTRODUCTION

- THE RATIONALE BEHIND SPI
- A SUCCESS STORY IN THE MAKING
- ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

CONCLUSION

CONCLUSION

☑ not a one-man show

Software Process Improvement
A Real-world Experience

Presentation for the
4th Annual International Quality Week Europe Conference
22 to 24 November 2000, Brussels, Belgium

Jacobus du Preez (jdupreez@arm.com)
Lee D. Smith (lsmith@arm.com)

ARM Limited, Cambridge, UK

www.arm.com

ABSTRACT

The paper relates a number of SPI successes in the Development Systems software group of ARM Ltd, a rapidly expanding, highly commercial small/medium enterprise that has consistently generated high levels of customer satisfaction and profit since its inception in 1991. It involves no “rocket science”. It is more about the effective reduction of process chaos under tight commercial constraints than about conventional SPI. It may appear to challenge some aspects of received SPI wisdom. Nonetheless, it highlights a number of key prerequisites to viable SPI, and involves principles which can be applied with good effect in most organisations.

INTRODUCTION

This paper is about successful software process improvement (SPI) in a rapidly expanding, highly commercial small/medium enterprise.

Software development in ARM began in 1991 with four people. Today, nine years later, there are more than 120 developers (a compound annual growth of around 50 %). The largest development team at ARM still employs only 15 engineers. Teams have – and need – their own cultures.

ARM has consistently generated high levels of customer satisfaction and profit, and our SPI efforts have always been constrained by these two factors. To ARM the two are strongly linked. We have never been allowed the luxury of "stopping the world" while we installed a new process and retrained engineers to use it. We have always had to *evolve* existing processes. The ultimate criterion that all improvement ideas must meet is unconditional profitability within one product development cycle.

In common with most small/medium enterprise software providers, our main problem has not been an absence of process or poor process. As we grew, our primary problem areas have been incomplete processes, patchy adoption of best local practice, and lack of process coherence among developers within and between development teams.

The "success story in the making" discussed in this paper involves no rocket science. It is more about effective reduction of process chaos under tight commercial constraints than about conventional SPI. It may appear to challenge some aspects of received SPI wisdom. Nonetheless it involves principles which can undoubtedly be applied with good effect in most organisations.

At corporate level, ARM is working at developing a Quality Management System that satisfies the ISO 9001-2000 standard (currently in final draft). Existing business processes are being mapped to the principles of the new QMS, designed to provide a framework within which the business can grow, improve, and satisfy customer needs and expectations. We foresee that our software development process will link up with that framework in due time.

This paper assumes an appreciation of the general context of SPI. Its discussion of theoretical issues beyond the real-world SPI account described is primarily aimed at positioning the account appropriately within that context.

THE RATIONALE BEHIND SPI

Loops Within Loops

That *quality* is an attribute of software no one would seriously wish to question. Just think "Y2K", for example. However, what in turn are the *attributes of quality*? Here we become conscious of finding ourselves in the domain of the abstract, where great minds do not think alike. Nonetheless, on the surface of the question there is a reasonable degree of likemindedness: quality software is – here we go – capable, reliable, installable, compatible, usable, efficient, testable ...

So far, so good (more or less).

The next question is obviously: what are the attributes of capability, reliability, installability, etc? Or, when faced with conflicting opinions for example on the attributes of compatibility and usability, on what grounds do we decide to work with one definition rather than the other? How do we go about deciding which definition reflects reality most accurately? And once we have decided, how confident are we that we have made the right decision? What level of confidence is adequate?

So far, so good? How far, how good?

Approaches have been put forward that allow quantification of quality attributes, as well as quantitative predictions of all sorts of quality-related effort and cost. This, of course, appears to be the answer, moving towards objective analyses, acceptable to all stakeholders. Yet who are these stakeholders? Does "acceptable" mean the same thing to all of them? How acceptable is acceptable enough? "Objective analyses" – how objective? How do we quantify acceptability and objectivity?

Hang on – have we got ourselves into a loop?

Shouldn't we abort this process, revert to gut feel, and just get on with the job? How much can we afford to theorise about these things before reverting to gut feel will not have become a more cost-effective way of achieving the same end? But whose gut feel? And which end? How do we know to what extent we have achieved this end? How do we establish the cost-effectiveness of our approach? How do we discern the wood from the trees here? (Or – hang on – was it the trees from the wood?)

And how many levels of nesting are supported here?

The Challenge

We do find ourselves in the domain of the abstract – at least that much is certain. The challenge is to find out how much else is certain in this web of nested abstract loops with which we could

end up busying ourselves under the banner of SPI. How do we distill some genuine, concrete improvement out of all the theory vying for our attention?

Oversimplification (or reverting to "gut feel") is not the answer. It does not do justice to the problem. If anything, it risks aggravating it. Overcomplication is also not the solution. It is too prone to infinite loops.

Where then is the commonsense, “good-enough” approach that will deliver the goods? Where is the balance? How do we make SPI *work* – for us, that is, showing meaningful results amidst the reality of our unique constraints?

The difficulty with abstract concepts is that we cannot compile, run and debug them the way we compile, run and debug programs. When our theory is flawed, there are no overnight build reports, or defect reports with screen shots and core or memory dumps, allowing comparatively quick resolving of specific issues.

Obviously this does not mean that the abstract is necessarily *vague* or *obscure*. The *vague* may be pointless in a technical context – try feeding your CPU vague notions about the on and off states of bits and see how far you get. What this does mean, however, is that there is a need for thorough analysis from the outset rather than the process engineering equivalent of "debugging by compilation". Process engineers cannot afford this practice – it takes too long, costs too much, and does too much damage when things go wrong.

Key Definitions

For the purposes of this discussion:

- A *process* is in essence a recipe for success. It is a sequence of steps for a given purpose[1]. The collective noun *process* denotes an integrated set of processes, all aiming at achieving a common ultimate end.

- *Software process* relates to the context of software engineering and is an integrated set of processes ultimately aimed at the creation of software products as per required feature set, budget and schedule in a given context.

- *Improvement* is measured progress along a defined worst-to-best axis. *Measurement* in the process improvement context is the positioning of instances of process along an appropriately defined worst-to-best axis.

- *Process improvement* is the *improvement*, as defined, of *process*, as defined, and involves controlled alteration of the process as part of an improvement strategy.

- *Software process improvement* is *process improvement*, as defined, applied in the software engineering context (or *improvement*, as defined, of *software process*, as defined).

- A *model* is a conceptual model rather than a process framework.

Engineering vs Process

In the above definitions we link process directly to the purpose of the activity it is designed to support. This implies that in our view process no more drives the software development effort than the tail wags the dog. *Process does not produce software; software engineering does.*

Software engineering, however, never happens without process, because it includes process. All but the most incompetent hackers follow some sort of structured development process.

The question is therefore not whether the dog has a tail or not, but rather how much of the tail is left. It is also not whether the dog wags the tail or vice versa, but rather whether we look at a dog or at some other strange phenomenon.

Software Process Goodness

He who says "improve" implies moving from "bad" to "good". What then is "goodness" in software process?

As stated, our definitions inextricably link the "goodness" of a given instance of software process to the purpose of the activity it is designed to support. That is, a particular manifestation of software process is only as "good" as the degree to which (or as the effectiveness with which) it *actually* supports the software engineering going on in the same context in the creation of software products as per required feature set, budget and schedule.

Software Process Reuse

If processes are recipes for success, then they are by definition designed for reuse. The fact that the same process will never be equally good in two different contexts does not mean that process reuse is necessarily a bad idea. Process reuse does not necessarily imply a “one size fits all” approach. Process reuse can be cost-effective when:

- the process is designed for a specific “market”
- production means (eg. process) is differentiated from production content (software)
- the process is kept lean, or limited to what is really necessary.

If the average process “goodness” registered during reuse along these lines is good enough, process reuse makes sense.

Measuring SPI

In our view, the main requirements for meaningful measurement of process "goodness" across a series of instances of the same process in the same context are *consistency* and *relevance* in the measurement approach. Relevance, that is, to what constitutes goodness in a software process, as defined above.

Formal audits and process frameworks are generally designed to promote among other things consistency and relevance in the measurement approach. However, meaningful measurement only happens where their *actual* implementation is consistent, as well as relevant to the development environment in question. In our experience these two conditions are not always adequately satisfied, with the result that the benefits achieved by implementing such formal approaches do not always outweigh the stress they introduce into the environment. Where we strongly suspect such a mismatch in advance, we would rather opt for a measurement approach which is less formal and less refined, but more consistent and more in touch with the reality of the environment.

The Essential Nature of SPI

There appears to be consensus in the industry today that competitiveness and high quality levels require the use of fitting processes[2]. At the same time, however, many serious software organisations are still battling with fundamental process issues (requirements capture, access to information, quantitative feedback on progress, time and cost estimation, etc[3]). Also, although successes are registered, a large number of SPI programmes still deliver little or nothing (i.e. are not even followed through to completion).

We stated that the challenge SPI poses is to distill, out of all the available theory, some genuine improvement in a particular development environment. In order to be able to meet this challenge, we need to:

- understand existing processes in that environment
- understand the issues that would be involved in specific changes
- select and introduce appropriate improvement ideas, and
- learn from experience.

The challenge is further to achieve as much of this as possible *up front*, that is, without creating situations out of which we may be required to backtrack. Finally, the challenge is also to achieve these things amidst the reality of ongoing competitive software engineering, that is, without requiring the world to stop turning or to start turning around us.

All this implies that the effect of failure or success in SPI tends to be exponential. More than in most other fields, success breeds success and failure breeds failure here, for a number of reasons:

- software is complex: the impact of getting PI right or wrong is just that much more significant in a software organisation than in a chewing gum factory
- process is central to the software development activity: it affects all team members
- process is designed for reuse: it affects multiple projects
- SPI has much to do with culture: it needs buy-in in order to get off the ground.

Two Crucial Factors

If it is true that the effect of failure or success in SPI tends to be exponential, two factors of paramount importance emerge:

- *The success/failure ratio.* Whatever our overall SPI strategy, it is crucial that we make a success of the *current* SPI initiative. The current initiative is the one thing which more than anything else will open doors for further sensible SPI. Hence we'd rather introduce say two SPI initiatives per year and give them the focus they require to guarantee success than introduce say ten initiatives and see half of them fail, because in our view those failures hold enough potential to derail our entire SPI drive. Or to put it differently, we'd rather carefully consider ten valid SPI ideas and implement only the two we are positive will succeed now. A high success/failure ratio ensures ongoing SPI, with all the potential that that holds for SPI becoming part of the culture of the organisation. This implies, among other things, that the younger an SPI programme is, the more crucial its need to register success.

- *Buy-in.* However sound or impressive the SPI initiatives we introduce, their genuine effect hinges on the level of buy-in they register. That is, in order to be successful, they need to become part and parcel of the software engineering activity *actually going on*. This is not the same as becoming part of the machinery *nominally*. It is possible for a process to be introduced, accepted and applied in such a way that it passes formal audits and yet is never "bought" by the engineering staff. In such a case, buy-in is low and compliance becomes a façade. This is the very sickness that often afflicts top-down quality engineering approaches. Compare two hypothetical processes A and B, aimed at achieving the same result. Process A can be implemented using a bottom-up approach, but is, in itself, only half as effective as process B, which needs to be imposed and policed. However, process A registers 90 % buy-in, whereas process B achieves only 10 %. We would prefer process A over process B, because it is likely to deliver more compound benefit in both the short, medium and long term.

Of course these key SPI success factors do not constitute SPI in themselves. That is, while it is difficult to imagine how SPI success can be achieved without them, these factors do not guarantee the soundness of the SPI initiatives for which they open the door.

The Case for SPI

SPI success has arguably become a key factor in the ability of any software organisation to remain competitive, and hence to survive, in the medium and longer term. If this is true, then we should as a matter of priority seek to fully understand the challenge SPI poses before using our limited process improvement budget for anything other than the most appropriate solution we can afford. The knock-on effects of failing to do that could be far-reaching.

A SUCCESS STORY IN THE MAKING

The Software Group

As stated in the introduction, software development in ARM began in 1991 with four people. Nine years later, the direct descendant of that group counts 15 and continues to produce our compilers and linkers as part of a larger group of about 40 engineers responsible for the complete software development toolkit we sell. Closely related products bring the total number of software engineers working within the business unit under study to nearly 60. Within ARM worldwide there are more than 120 software engineers. Software development activity has increased by around 50 % each year. Shipments of the software development toolkit have increased 100-fold in these nine years (approximately 70 % CAGR).

Growth Pains

This sort of commercial success poses a number of difficulties for conventional approaches to software development process and its improvement:

- There is never enough effort. Recruitment always lags need. Around 25 % overload is endemic.
- Management structures are lightweight, dynamic, and impermanent. Informal reorganisation happens frequently. Major formal restructuring becomes necessary every two to three years.
- There is little visibility of the future. The next quarter may be 90 % predictable, but the one after that would typically be only 60 % predictable, and the next one down the line only 40 % or so. Development projects, however, span three to eight quarters. Most projects need to be replanned more than once.
- Within the general constraint of alignment with company business objectives, developers are allowed to use their own discretion when reacting to customer requirements. In fact, they are encouraged to take a customer-oriented approach, and the company prefers to rely on their judgement rather than introduce measures that might constrain initiative or reduce responsiveness to customer needs.
- High growth rates – in personnel and in product volumes – can rapidly change *the most important problem to solve next*. Process problems tend to be moving targets, and trying to improve processes sometimes feels like shooting rabbits from a speeding train.

Full-time SPI

To date we have never seen our way open to introduce into this environment an SPI programme based on carefully documenting existing practices, then closing the gap to industry-wide best practice, etc. We can conceive of such an SPI approach being effective when development is chaotic within an orderly environment, that is, where the challenge is to impose order within a stable environment. Our problem, however, is to order development in a chaotic environment! How does one achieve that?

J du Preez, L D Smith

Of course we agree with much of what frameworks such as CMM, ISO9003, etc propose as sound software development practice. We have always understood, for example, that we would never survive unless we explicitly project managed the production of software products. From inception, we have used automated build and test techniques, as well as version-controlled source code. We have always been able to build products in a repeatable manner, and trace changes in detail. These capabilities are just essential entropy reducers without which there is no hope of orderly development in an otherwise chaotic environment.

The growth in numbers, resulting in various teams working in parallel, obviously did nothing to reduce the chaos. At the same time, software development was becoming increasingly important to overall company business objectives, and we realised that a more specific and proactive focus on development process was required if we were to retain the competitive edge we had in our market. Among other things, this resulted in the appointment of a full-time internal SPI "consultant".

A Suitable Climate

Following his interview at ARM in mid 1998, there were specific reasons why Du Preez believed that the software group at ARM constituted as ideal a climate for “growing” real SPI as any he was likely to encounter:

- The manager of the software group came across as a down-to-earth, commonsense man, capable of perspective and decision making.
- The team was growing and the team size was right: the growth necessitated process upgrades, and yet they were small enough to be open to new improvement ideas.
- The team managers had burnt their fingers once or twice with premature releases and were keen to reduce the risk of further similar incidents.
- Opinions of acquaintances who knew ARM directly or indirectly painted a picture of a non-bureaucratic company with comparatively low levels of internal politics.
- The company had been registering phenomenal growth, as regards both head count and revenue, so spending a few pounds on SPI would hopefully not be a big deal.
- The team was producing software which was actually being sold and used. Theirs was not an artificial environment: what was being produced today actually mattered tomorrow.

The company profile sketched here should be seen in context. The intention is not to suggest that genuine SPI will not "grow" in companies with a different profile, but simply to give an account of the situation as it evolved.

Three Dimensions

When Du Preez joined the ARM software team in March 1999, he was thinking: *Be careful – they must be doing something right, otherwise they wouldn't be successful at selling their software.*

How does one go about improving process in such a situation? Most of the team were right in the middle of a sizable project – never a good time to want to change the way they work. However,

this provided a good opportunity to assess the context in the three dimensions which he believed can and must be applied to any SPI programme:

- *The people dimension, defined in terms of the actors with whom the process improver needs to interact (covering all levels – everyone affected by the process to be improved).*

  Process improvement is not defining an "improved" process, presenting it to the team and then "making everyone do it". Such an approach excludes the people dimension from the activity, which cripples it, because people are the very source of energy that fuels it.[4]

  Process improvement is:
  - thinking carefully about the existing process and any potential for improvement
  - convincing process stakeholders that a better version of the process is both desirable and feasible in the current context
  - defining the improved version with their help (they know what is going on on the ground)
  - *facilitating* the implementation of this improved version
  - keeping the door open for further improvement.

  None of these phases is possible without meaningful interaction with process stakeholders.

  This does not imply that process improvers must always keep everybody happy – it implies that without successful interaction with key people at all levels they cannot do their job.

- *The dimension that defines the sequential flow of the process, running more or less along the so-called software development life cycle.*

  This dimension requires not only an understanding of software development life cycles, but also the ability to accurately assess particular instances or adaptations of software development life cycles, so as to be in a position to appreciate the potential overall impact of improvement ideas.

- *The worst-to-best dimension (see above) allowing the relative positioning of particular instances of process in terms of their "goodness".*

  This dimension requires a worst-to-best axis definition that is appropriate for the context, as well as the positioning of particular instances of process on that axis. This in turn implies the achievement of meaningful measurement mechanisms enabling the assessment of the overall level of support which instances of process provide to the engineering activity, as described.

As part of his initial assessment of the context in terms of these dimensions, Du Preez threw a number of ideas into the software group purely to see what would emerge. For example, a rather formal sounding suggestion for a full-blown three-year improvement programme, which was ignored by all recipients, and which helped to shape his subsequent approach.

A Starting Point

The first real opening presented itself when it became clear that no serious test coverage tracking was being done. A plan was agreed and implemented with the help of the SQA team to

track the extent to which test phases were respectively covering application components, interfaces, functionality, user interface elements, code, documentation, recovery from equipment failure, processors, architecture types, and operating systems. The idea was not so much to show up inadequate test coverage – helpful as that may have been – but to *highlight the need for good quality specifications* on which to base tests and against which to measure coverage.

This, in conjunction with a slide show presentation to the group about good practice and hands-on participation in system testing while waiting for the project to finish, provided opportunities to get acquainted with the team and the situation, and to consider the next step.

A Long Shot

Before too many decisions were going to be taken about the next project, Du Preez submitted a major improvement proposal to the software group. This was risky, because:

- the proposal laid it on thick, involving comprehensive quality tracking throughout the life cycle, using a utility which didn't exist as yet as a kind of all-encompassing framework
- initial discussions about the proposal triggered warnings that the group would reject it because they had grown allergic to new utilities after a previous bad experience
- attempts to gauge management support before the gathering revealed that one out of four managers was negative, two were neutral, and one was positive
- Du Preez had often in his career been accused of being "binary" about quality, and had to ask himself whether it was wise to table such a radical proposal at such an early stage.

However, he was thoroughly convinced of the sense of the proposal, and decided to go ahead.

A Turning Point

The result was an overwhelming vote of confidence in the proposal. This constituted a turning point in more than one way. Firstly, the manager of the software group bought the proposed idea, albeit with reservations. He told the company Quality Steering Group about it and they asked for a repeat show, following which they requested a functional specification for the utility. Secondly, the true colours of the team became apparent, as discussed below.

A minor restructuring became necessary soon afterwards, and one of its side effects was an agreement worth describing. The SPI consultant now reports directly to the software group manager. His brief allows him to question any existing practice and suggest any improvement idea he sees fit. He has no interests to protect other than improving development practice. At the same time, the group is under no obligation to follow his advice. He is a consultant – he is there to provide advice, and if he wishes to see his advice implemented, he needs to effectively “sell” it.

The Mind of a Programmer

Many roles contribute to success in software development. Various contributions are crucial at times. However, if one role is to be singled out without which no commercial software is ever

shipped, it is that of true programmers – those members of the broader software engineering community whose experience and skill sets make all the difference when complex and/or unexpected coding challenges present themselves.

The nature of the job these programmers are required to do conditions their minds to think logically. When confronted with an "improved" process, the first thing they think is: *why*? If a good reason can be provided, they will support it. If no good reason can be provided, their support will be lacking, overtly or covertly, to the very extent and for the very reason that they are valuable to the project.

We believe that the Quality Tracker proposal received an overwhelming vote of confidence from the software team because it was based on good reasons, and because the links between the model and the reasons held water as explained.

The Quality Tracker

The proposed utility — generally referred to as the Quality Tracker — has not yet been produced and may never be fully implemented as originally intended, for reasons explained further down. The concept behind it, however, underpins all SPI we have done since.

In brief, the Quality Tracker is designed:

- to interface with existing systems rather than replace any of them
- to use the same product breakdown as other systems (eg the defect tracking system)
- to ensure, in as objective and automated a fashion as possible, the capturing of all development progress which has a bearing on quality
- to reflect quality progress in terms of mini-milestones that link directly to master copies of associated information, and which have attributes such as priority, owner, weight, entry criteria, exit criteria, etc
- to reflect the entire product development life cycle using these mini-milestones
- to allow self-regulation (for example following lessons-learnt sessions) in that mini-milestones are concensus-driven
- to manage dependencies between mini-milestones pertaining to the same module
- to cater for product hierarchy (linking modules into subsystems and subsystems into a systems) and manage dependencies between product elements at all levels
- to provide visibility of actual progress by maintaining multiple pointers per module (multiple pointers to cater for tasks executed in parallel on the same module, with pointers indicating mini-milestones currently being addressed)
- to use weighting of both progress steps and of modules in order to calculate overall quality progress achieved per product at the click of a button
- to be flexible by allowing a compulsory/non-compulsory flag to be set for each mini-milestone, and by then permitting manual overriding of non-compulsory mini-milestones
- to maintain and use progress and exception history to calculate confidence levels per product at the click of a button.

The Inspection Process

The Quality Tracker slide show presented a concept. The actual mini-milestones remained to be defined. This resulted in a discussion about what would be more appropriate during a first phase: a coarse granularity, including only key milestones but covering the entire life cycle, or a fine granularity, including all appropriate mini-milestones but covering only a limited section of the life cycle. In the end, the approach was adapted to match an acute and generally recognised need. The fine granularity approach was to be applied to only one life cycle element: Specification of User-visible Behaviour (which we call SUBs).

The reason for selecting SUBs rather than anything else was two-fold:

- it would provide useful and better quality early input to the documentation and test teams
- it was likely to highlight flaws in a number of related processes and products, both upstream (eg requirements specifications) and downstream (eg low level designs) from SUBs.

Another slide show was lined up to sell the concept of limited-effort inspections on SUBs. This resulted in a focused, measurement-oriented inspection process, deliberately designed to be "lean and mean". This process:

- is applied at the point of sign-off, that is, after a technical peer review and at the point where the author considers the SUB to be finalised
- is viewed as a quality sampling process (applying cut-off points as regards time spent) rather than as a process designed to cover the total product
- is not compulsory, but is a service provided upon request

Apart from the above, the inspection process is not exceptional: it requires participants to work with a specific primary role or focus in mind with a view to optimising the effectiveness of the process, and it applies a number of tried and tested rules to ensure its own survival.

The following roles are used:

- Author – *has written the specification (answers questions when called upon to do so)*
- Instigator – *whose prior work has given rise to the specification currently being inspected*
- Peer – *colleague of the author, with essentially the same skill-set and experience*
- Customers – *whose work depends in some way on the specification being inspected*
- Tester – *SQA engineer due to test the product based on the specification being inspected*
- Moderator – *ensures that participants confine themselves to their roles and respect the rules*

The following "tried and tested" rules are applied:

- *Inspection training beforehand (typically a one hour slide show presentation followed by a discussion)*
- *The inspection consists of two parts: 1 – at participants' convenience; 2 – common, bug-logging session*
- *No changes to work between the point of submission for inspection and the inspection itself*
- *The second or common part of the inspection never lasts longer than one hour*
- *Nobody ever spends more than one manday per week doing inspections*
- *No extended discussions on how to fix a defect*
- *No extended discussions on whether or not a defect exists*
- *Authors do not defend their work, except in response to questions*
- *No extended technical explanations*
- *Style issues are not to divert focus from the substance of the job*

Success Breeding Success

Just at the right time – after the inspection process had been agreed and defined, and before the first SUB would be ready – Du Preez's neighbour requested an inspection on a (non-SUB) specification he had been producing. This presented an ideal opportunity to try out the process.

In spite of the fact that the specification had been peer-reviewed and turned out to be of above-average quality, the inspection still uncovered a surprising number of valid defects. Participants were positively impressed with a clear demonstration of the value of early defect detection. The inspection process was off to a good start.

As the effectiveness of the process became apparent (see below) management started expecting all SUBs to be submitted for inspections. One day, for example, the group manager was overheard stating the case for inspections to a team leader who had phoned to ask for an exception on one of his SUBs.

As anticipated, close scrutiny of SUBs showed up flaws in requirements specifications. Soon requirements specifications were also being submitted for inspection. Previously it had always been difficult in our environment to secure feedback from requirements stakeholders during peer reviews, but the inspection process secured impressive stakeholder representation for requirements specification inspections, resulting in excellent visibility.

The process became so popular in the end that all sorts of products were being submitted for inspections, from low-level specifications to the inspection process document itself.

Increasingly participants in inspections were coming from other teams in the business unit. Some of these started prompting their managers to adopt the process as well, and currently the process is being introduced BU-wide.

Requirements Management

The inspections conducted on requirements specifications stretched the inspection process and showed up a few shortcomings, but resulted in clear benefit and became a forceful demonstration of the need for improved requirements management. It highlighted for example:

- the need for a common requirements specification template
- the need to enable users to pinpoint complex information quickly and easily
- the need to update individual requirements and keep them current throughout the life cycle.

Of course there are numerous reasons why all reasonable attempts should be made to optimise the process of capturing and managing requirements – this is not the place to describe them in detail. The challenge is to render enough of these reasons apparent to ensure adequate momentum for the introduction of improvement ideas to succeed.

From a process improvement perspective, we now had the wind from behind. We were in a position to start putting the horse before the cart. Just about every SPI initiative we could foresee for the medium term was part of the cart in the horse-before-the-cart analogy, and

without the horse of sound requirements engineering we couldn't see how that cart was going to get anywhere.

Our next step was to assess requirements management solutions available on the market. Two questions needed to be answered:

- is it a good idea to buy a requirements management tool?
- if yes, which tool would be most suitable?

Three tools were shortlisted for evaluation on a trial basis. A tool was selected, and three licences were purchased, with a view to evaluating it in depth. At the time of writing we are embarking on a pilot project in which traditional, document-based requirements capture is being replaced entirely by tool-based requirements capture. The tool will also be used to manage a limited number of requirements objects throughout the development life cycle, covering a few typical user scenarios.

A Success Story

To date, we only have limited and crude data available for assessing the comparative effectiveness of measures introduced over the past eighteen months. The table below gives a crude but conservative comparison between the defect find rate achieved by inspections during the current project and the overall defect find rate during the previous project.

Previous Project: All Testing *(comparatively major toolkit upgrade)*	Current Project: Inspections Only (first 13 weeks) *(comparatively minor upgrade of same product)*
1500 mandays' test = 1500 * 6 manhours = 9000 manhours *(excluding developer test time)* 1714 defects total *(including defects logged by developers)* **0.2 defects/manhour**	180 manhours (inspection participants) + 500 manhours (Du Preez full-time) = 680 manhours *(using not 6 but 7.5 manhours/day)* 417 defects total *(found early in life cycle, i.e. cheaper to fix)* **0.6 defects/manhour**

This comparison seems to indicate that the inspection process was at least cost-effective, especially considering that requirements and design defects found early cost much less to fix.

However, in our view there are a number of further reasons for qualifying the process improvement drive described as a "success story in the making":

- *Its Success/Failure ratio.* We have yet to see an improvement idea fail.
- *Its Popularity.* Each success "breeds" more success. The Quality Tracker presentation played a key role in preparing the way for the inspection process, and the inspection process played a key role in opening up the opportunity to improve requirements management practice.

- *Its Soundness.* It is outside the scope of this paper to explain in detail why the main initiatives introduced represent good development practice, but in our view they could hardly have been more in line with proven, sound practice.
- *Its Proactiveness.* The initiatives described lay a solid foundation for further SPI over the medium term, and are indispensable to sound practice during later life cycle phases. Also, the work described has made a positive impact on the way engineers think about SPI, promoting an awareness and an appreciation of the issues involved in proactive quality engineering, and leaving the door open for further work.

In addition, consideration is already being given to the idea of broadening the scope of the current SPI drive to cover the entire Development Systems Business Unit. Du Preez has recently been tasked to capture and document existing practice across the BU with a view to analysing the situation and investigating opportunities for improvement.

Next Steps

The first thing we need to achieve now is success with the tool-based requirements management pilot project. As stated, we believe that that will give us a horse with which to pull the cart of all further SPI we have in mind.

Following a successful pilot project, we hope to roll out the tool to the whole team. We plan to use the pilot project to define a customised training package, designed to minimise the hassle factor once the tool lands on people's desktops.

Full application of the tool implies upgrading our development process to allow requirements traceability across the entire life cycle. We plan to rethink and redefine the development process where necessary in conjunction with the definition of the training package. This exercise will also help focus our attention where it belongs for subsequent work.

Once we have fully applied the requirements management tool, we also need to ensure its integration with existing systems, such as the defect tracking system, our version-control system, and the ARM project management system.

Some of the ideas of the Quality Tracker will have been implemented by the requirements management tool. After integrating it with existing systems, we plan to revisit the Quality Tracker concept with a view to implementing an adapted version of it.

Our ultimate objective is to see proactive quality engineering become engrained in the underlying fibre of the team, enabling full and ongoing compliance with ARM's stated Quality Policy of satisfying customer needs and expectations while sustaining and developing business growth.

ESSENTIAL ATTRIBUTES OF AN EFFECTIVE SPI MODEL

In the light of our experience, we would suggest that a few key attributes distinguish effective SPI approaches – primary characteristics without which the thing will not fly.

Orientation

An effective SPI model has direction. Like a magnet, it aligns itself with the objectives of the organisation. Its initiatives are always part of an overall design that serves stated goals directly.

Example. The slide show used to present the Quality Tracker concept carefully took into account the objectives of all key stakeholders: group management, project management, SQA, Documentation, Engineering and Support. We believe that the overwhelming vote of support it received was a direct result of the fact that the links between these objectives and the various elements of the Tracker concept could be made clear. All subsequent SPI efforts have been in line with the Tracker concept.

Popularity

An effective SPI model is used as intended and is respected and owned by its intended users. The less a model is used, the less its effect, and the more investment is required to secure its usage.

Example. The inspection process introduced is an eminent example of a popular process. It is a stringent process, and was intended to be applied only to SUBs during its first project. However, within the duration of that first project, the process sold itself to further life cycle phases (requirements, low-level design), to other projects, and to other groups within the same business unit. Its popularity played a key role in opening the way for a further major initiative: the introduction of a requirements management tool.

Clarity

An effective SPI model is understood by those who use it. It can be explained to them and it makes sense to them. Any lack of clarity is superficial and removable. The intent of the model is clear.

Example. Throughout the SPI drive described, we took care to ensure and confirm that concepts put forward were clearly defined. Using slide show presentations to the entire (distributed) team, as well as a well-documented intranet page, we conveyed ideas and invited comments, always ensuring that key team members understood what we meant and that any concerns raised were taken seriously. SPI without a clear vision cannot expect to command the respect and support of software engineering staff.

Appropriateness

Both the what and the how of an effective SPI model is suitable for the context it seeks to serve. The support it provides matches the real need of the situation. It wastes nothing; it lacks nothing.

Examples. The Development Systems software group has a "keep it simple", non-bureaucratic culture. The word "formal" scares them. At the same time, team members are proud of their contributions. Hence the initiatives we introduce rely little on bureaucracy and much on trust. Further, the group is, and has been ever since its inception, in the business of actually delivering software and making a profit out of it. Hence the initiatives we introduce are calculated to show a return on investment within one project cycle.

Flexibility

Like a dog's tail, a successful SPI model is flexible. Like aircraft wings, it does not break off when required to bend in the interest of the environment it serves.

Example. Some of the ideas which Du Preez raised early on in his career at ARM were shot down, almost as soon as they were voiced, by prominent team members who perceived them to be irrelevant to their particular interests. A rigid approach at the time would have been disastrous. Flexibility (without losing perspective – bending without breaking) ensured ongoing SPI. In the meantime, some of those ideas have been implemented and the same team members have become supportive of SPI initiatives introduced.

Automation

Cost-effective exploitation of automation opportunities characterises successful SPI models. SPI models that promote automation tend to reduce bureaucracy, facilitating more and imposing less.

Examples. The main reason why requirements do not remain current throughout the development life cycle is lack of automation. In our view, opportunities for medium term progress are limited without automating the requirements engineering process. Also, the Quality Tracker concept is an attempt at automating centralised quality status tracking. Manual tracking could never provide the same levels of visibility on real progress, no matter how hard we tried, and the knock-on effects of this lack of visibility are substantial.

Simplicity

Successful SPI models "keep it simple". As simple as possible, that is – as specific as is necessary, without being over-specific.

Example. When the first inspections happened, pass/fail criteria had not yet been finally agreed. We decided to base pass/fail decisions on consensus among participants for the time

being. This worked so well that we decided to “keep it simple” and to forget about the idea of preagreed pass/fail criteria. Quite a few specifications failed inspections, but the validity of decisions reached has never been questioned. On the other hand, rigid application of the criteria originally discussed would have led to dissatisfaction in some cases.

Coherence

SPI which is consistent with a well thought-through overall SPI strategy based on sound development practice is more likely to be successful on the whole.

Example. The soundness of making reasonable efforts to contain defects within life cycle phases cannot be denied. A focused inspection prior to sign-off (before making products available to subsequent phases) is one of the most effective ways of uncovering defects within the phase during which they are introduced. The implementation of a clearly defined, generic inspection process (which can potentially be applied to any product during any life cycle phase) is totally consistent with the principles of early defect detection and phase containment of defects.

CONCLUSION

Getting SPI right is a tricky business. If the account described in this paper has taught us one thing, then it is that successful SPI is not a one-man show. The recipe which appears to be working for us is a balance between:

- a non-interfering, yet strong, informed and supportive management style
- an open-minded team, trusted to think for themselves and taken seriously when they do so
- a dedicated, pragmatic focus on SPI, not imposed but born out of conviction.

In fact, it strikes us that even the most accomplished process model would remain largely ineffective if the above balance is lacking.

The processes described in this paper may be regarded by some as comparatively immature, and perhaps rightly so. We do not yet formally comply with any known quality framework, and we do not yet use any complicated statistical models. Yet the initiatives introduced are in many respects an SPI dream come true: they are eminently sound, popular and cost-effective. The balance between process and engineering as a whole appears to be healthy, and the door is wide open for further SPI along similar lines.

After exposure to all phases and various models of large-scale software engineering, Jacobus du Preez became increasingly fascinated by the challenge of predictably engineering quality in software, in spite of the particular dynamics of the industry. His search for a climate where genuine SPI would "grow" led him to the Development Systems software group of ARM Ltd, where he currently works as a full-time internal SPI consultant.

Lee Smith has been with ARM since its start-up days, and has been involved in its software process improvement initiatives for more than five years. Smith is also co-author of "Challenges in Cross-development", an article published in the Jul/Aug 1997 IEEE Micro.

---

[1] IEEE Std 610.12

[2] Jalote, Pankaj; "CMM in Practice: Processes for Executing Software Projects at Infosys"; Addison-Wesley, Reading, Massachusetts, 1999; p xi

[3] McGuire, Eugene G.; "Worst Practices: A Field Report on Software Development Weaknesses"; Computer Science and Information Systems, American University, Washington, D.C.

[4] Bach, James; "Microdynamics of Process Evolution"; IEEE Computer Society; Feb 1998

[4] Goodhew, Peter; "Achieving Real Improvements in Performance from Software Process Improvement Initiatives"; ESPI Foundation, Ref AA008P; p 6

QWE2000 Session 6I

Mr. Olivier Denoo [Belgium]
(ps_testware)

"Usability: A Web Review"

Key Points

- What is usability testing?
- Why usability Testing?
- Objective versus subjective usability criteria.
- Review and results of 30 websites concerning their usability.

Presentation Abstract

Despite the efforts of Internet gurus like Nielsen or others, a large number of web sites are still not usable. Some could argue that usability is a fuzzy concept widely depending on users tastes, that you cannot satisfy everybody and that rules can be contradictory...it is true!

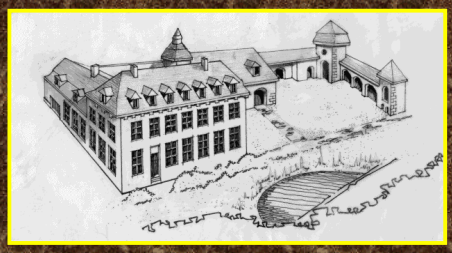
With a couple of Internet-minded users, we reviewed the progress of the historical usability study on Sun Microsystem site and, though everybody acknowledged the drastic progress that had been made, some were still not happy with the final result. Does it mean that the web site is not usable?

No! Since we are not Sun users and as the main critics were addressed to graphical contents and icon choices which are among the most subjective ever, our remarks are a bit out of scope, or at least second range. And that is a bit the problem, because some aspects of usability seem to be rather subjective, some web sites builders take the tempting opportunity to neglect them, pretending that usability testing is so partial and senseless that it isn't worth the effort (read price).

About the Speaker

Olivier Denoo is a consultant active in the Business of Structured Software Testing since 1997. He was the key-developer of ps_testware's Y2K testing techniques. Nowadays, he still is a highly respected trainer of this methodology. Continuously looking for new challenges, Olivier started to investigate the possibilities of website and e-commerce testing and became in charge of ps_testware's e-commerce/WWW-testing knowledge base. Currently, he is working in a project at one of the largest Belgian Telecom companies.

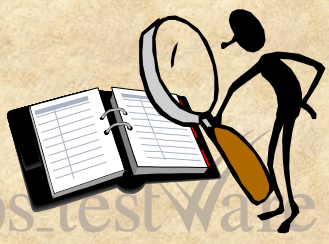
QWE2000 - Usability Testing



A review of current errors

Agenda

- Introduction
- Why testing usability?
- Methodological approach
- Usability and the web
- A quick review
- Conclusions
- Questions



ps_testWare
Software Testing Services

What is Usability?

What is Usability? (Jakob Nielsen)

- **Usability is the measure of the quality of the user experience when interacting with something**
- **The usefulness of a system is determined by two components:**
  - **Utility: Does the system do anything that people care about?**
  - **Usability: Can the user use the system and can he or she do so effectively?**
    - » **Ease of learning**
    - » **Memorability**
    - » **Efficiency of use**
    - » **Error type and frequency**
    - » **Subjective satisfaction**

ps_testWare
Software Testing Services

ps_testWare
Software Testing Services

Why Testing Usability?

Why is it a Conditio sine qua non?

Why Testing Usability? The Facts

- **Developers sell their products to real users**
  (machines, technology and systems don't buy)
- **Easier is better when you have the choice**
  (why bother with complexity when easiness is knocking at your door)
- **Usability is very critical in some areas**
  (Especially in E-commerce and communicating applications)
- **Consumer is king!**
  (You have to adapt to their way of working and not the contrary)

ps_testWare
Software Testing Services

Why Testing Usability? The Goals

- Historical records and knowledge base

- **Minimise the costs (hotline…)**
- **Increase sales and the probability of repeat sales**
- **Minimise risks (e.g. feeling secure on the Web)**
- **Acquire a competitive advantage as usability has become a product separator (e.g. Linux)**

Intermezzo

Why is the web different?

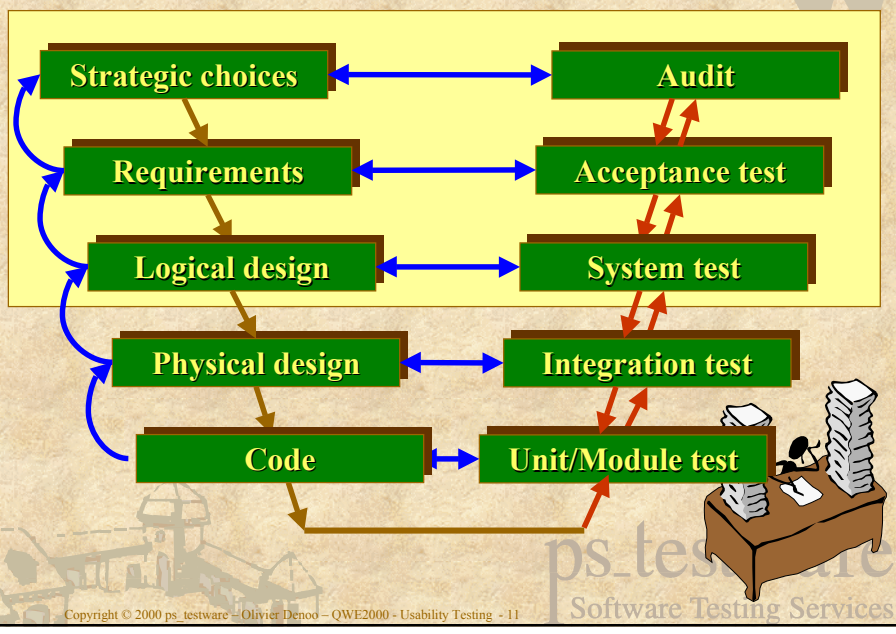
Web vs. Client/Server Applications

- **Target Market** (more than ever consumer rules)
- **Media particularities** (specific constraints)
- **Usability** (even more business critical)

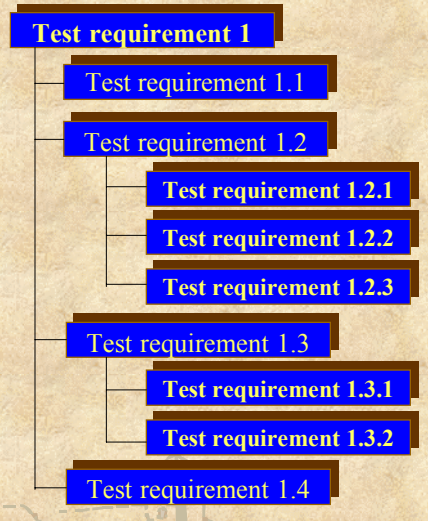
- Users do not read, they scan "no time to read on-line"
- Love me or leave me "offer is wide, no constraint accepted"
- Download time is an issue "when you pay the bill"
- Feeling secure is an issue when security is "extra muros"
- Users have low error tolerance
- Users are NOT "educated" some don't even want to
- Target market is fuzzy or unknown
- Portability and standards are issues

Methodological Approach

The V-model :



Test Requirements Hierarchy



Test Requirements Hierarchy:

- Structured hierarchical set-up of test requirements
- Process-driven (related to the business)
- Trace every possible step to the business goal (from business to functionality)

Setting-up Business Process Thinking

- Web applications are communicating applications
  - Market should be correctly targeted (WHY? WHO? HOW?)
  - Corporate image should be clear & relevant
  - Business processes should be clear, easy & user-friendly
  - Confidence has to be built
  - Business transactions should be understandable and under (user) control
  - …/…
- Web site has to sell itself/ application is already sold : consequences for testing
  - USERS need to be more involved
  - BPT is a solution

Techniques

- Expert review
  - done by experts
  - mostly find "objective" usability defects
  - adopt either expert's or supplier's point of view
  - TRH, Business Process Thinking, scenarios
- Guided (business) scenarios
  - done by users (real users, target groups…)
  - coached by suppliers / experts
  - mostly find business process defects
  - adopt users' point of view
  - Business Process Thinking, scenarios
- Recorded or logged sessions
  - done by users (real users, target groups…)
  - logged by suppliers / experts
  - mostly find subjective defects
  - adopt users' point of view
  - ad-hoc

ps_testWare
Software Testing Services

Our Review of Common Defects Found over the Web

Large companies make mistakes too ...

Important Warning and Remarks

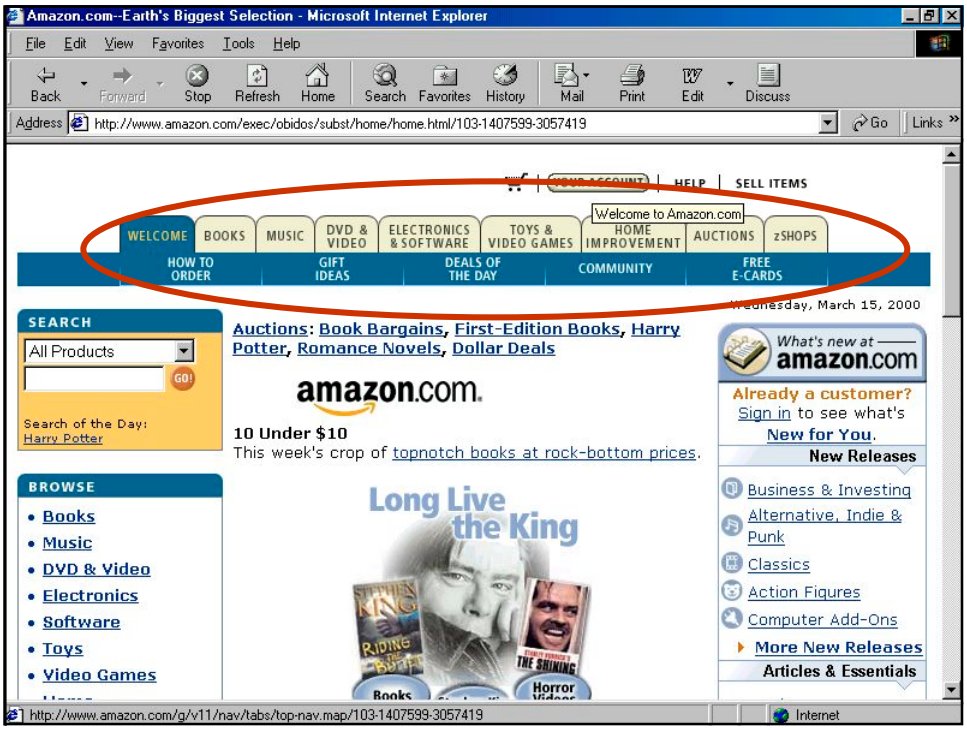
- **Errors shown here were observed "live" in March 2000. Some are or could be already corrected now.**
- **Everybody makes mistakes: large companies web teams...but also us, and their competitors as well**
- **There is no relationship between the quality of a web site and the quality of services and products advertised on that web site.**
- **We respect corporate images, copyrights and trademarks**
- **This presentation has no commercial goal, it is for Educational purposes only**
- **Examples were chosen randomly… but we can find more elsewhere if you ask us to do so.**
- **It is only our (subjective) opinion, though we qualify the defects as being « objective »**

ps_testWare
Software Testing Services

ps_testWare
Software Testing Services

Missing ALT contents

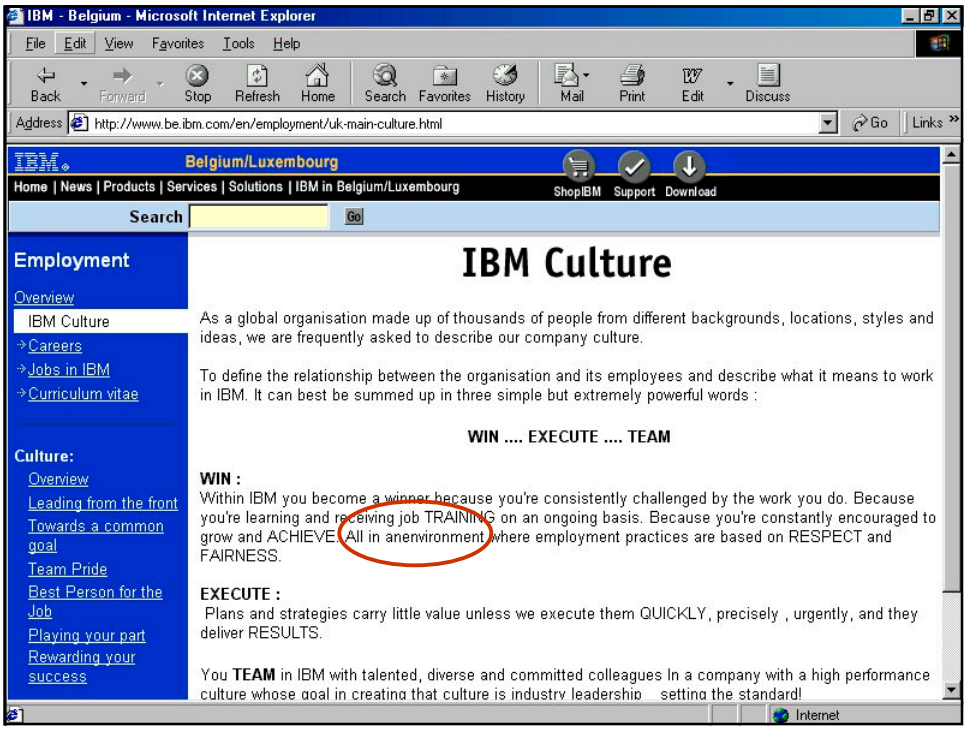
No Refuges – Ban Text Mode
Mapped Images Have no Memory



ps_testWare
Software Testing Services

Spelling Mistakes

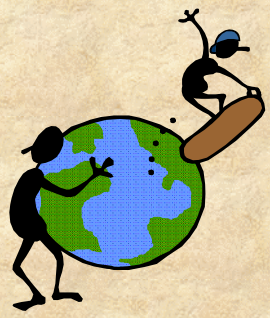
Little Attention to Details

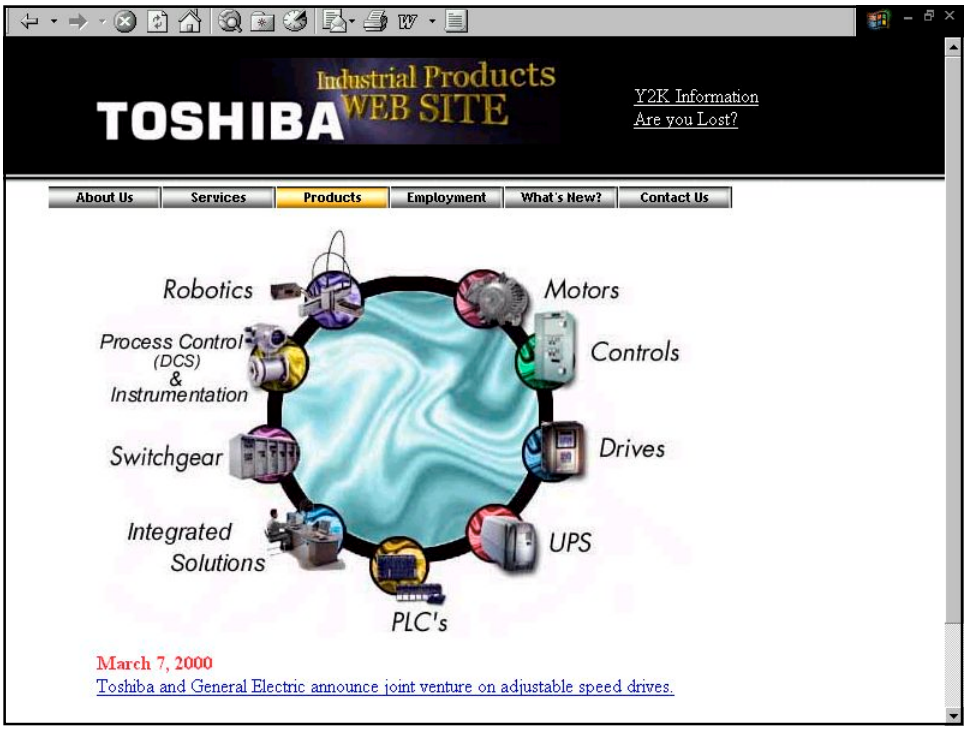


ps_testWare
Software Testing Services

Unclear Corporate Image

Lost in space





ps_testWare
Software Testing Services

Heavy Graphical Contents

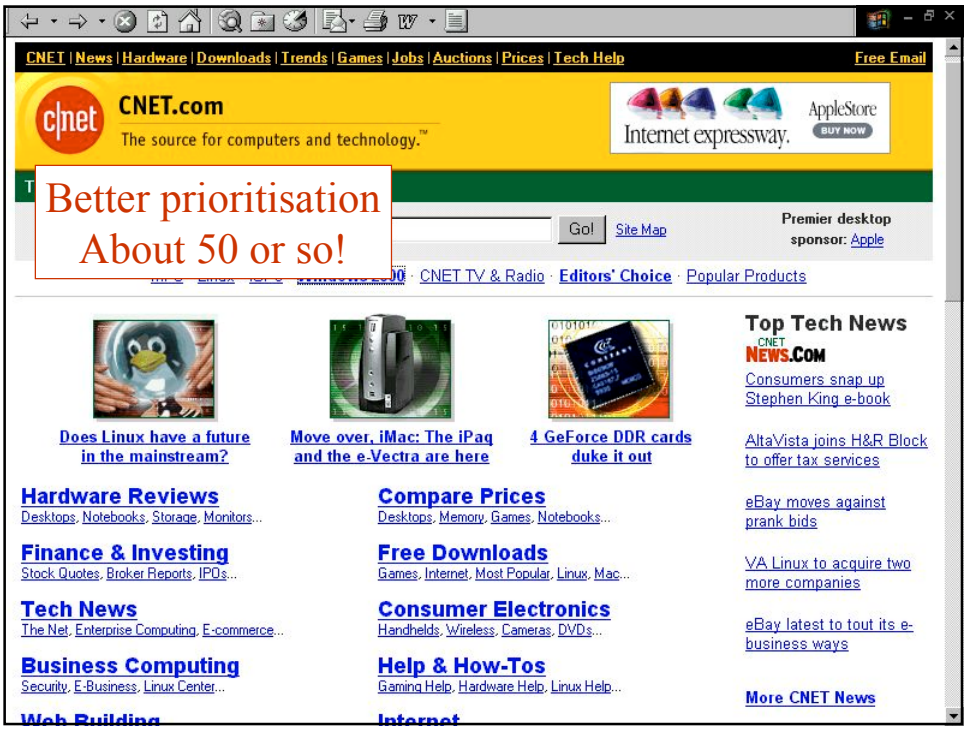
Please hold on, the first icon will appear in a minute or two!

ps_testWare
Software Testing Services

No Prioritisation

If everything is important…

nothing is important!

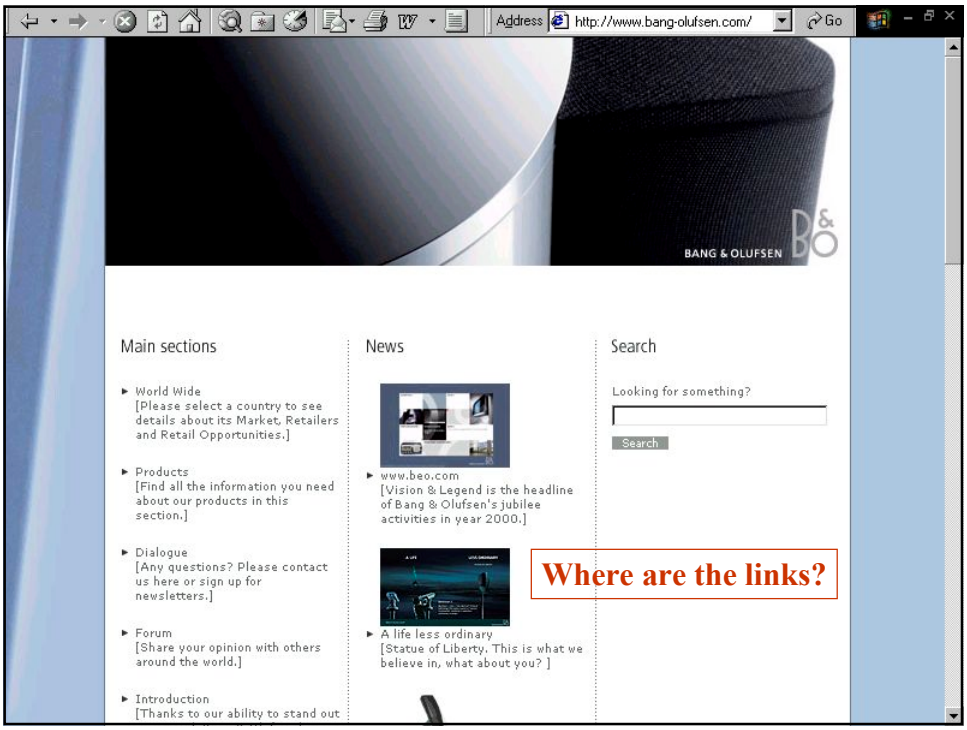


Better prioritisation
About 50 or so!

ps_testWare
Software Testing Services

No Breadcrumbs

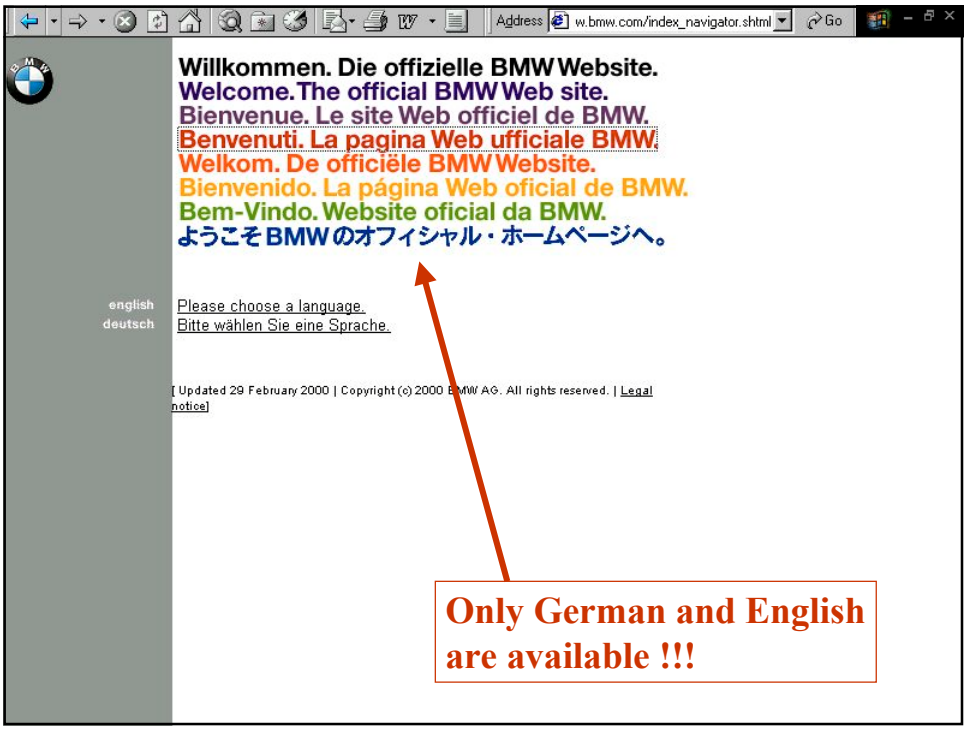
Sorry Gretel, we can't find our way back home I'm afraid!



ps_testWare
Software Testing Services

Translation & World Culture

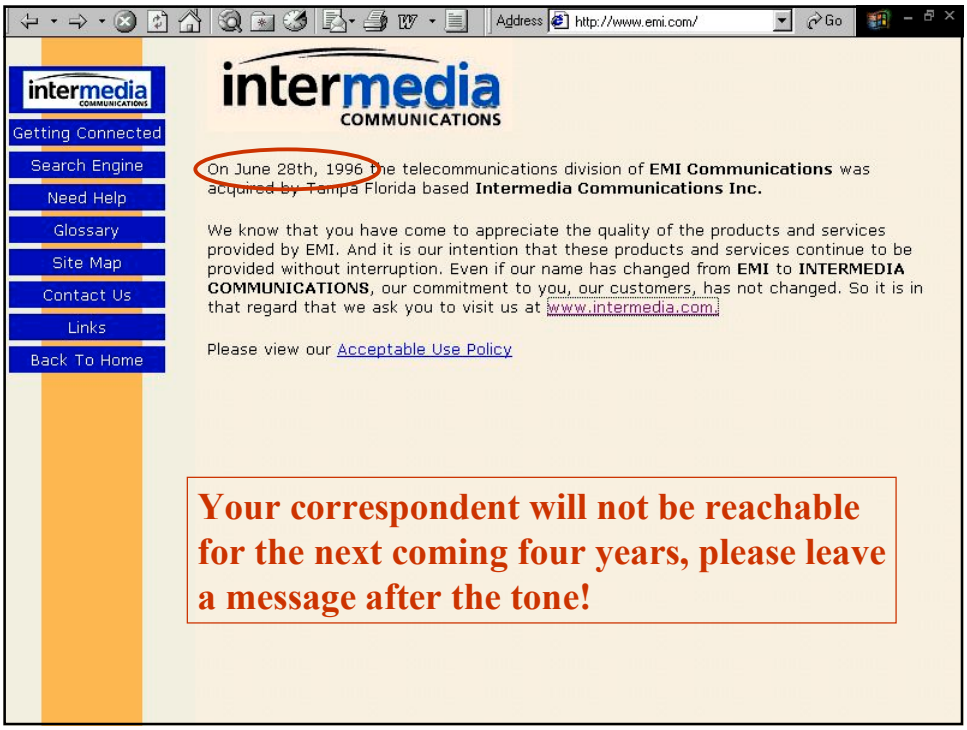
Je Speak Nederlands; Du Parla Español?



ps_testWare
Software Testing Services

Outdated References

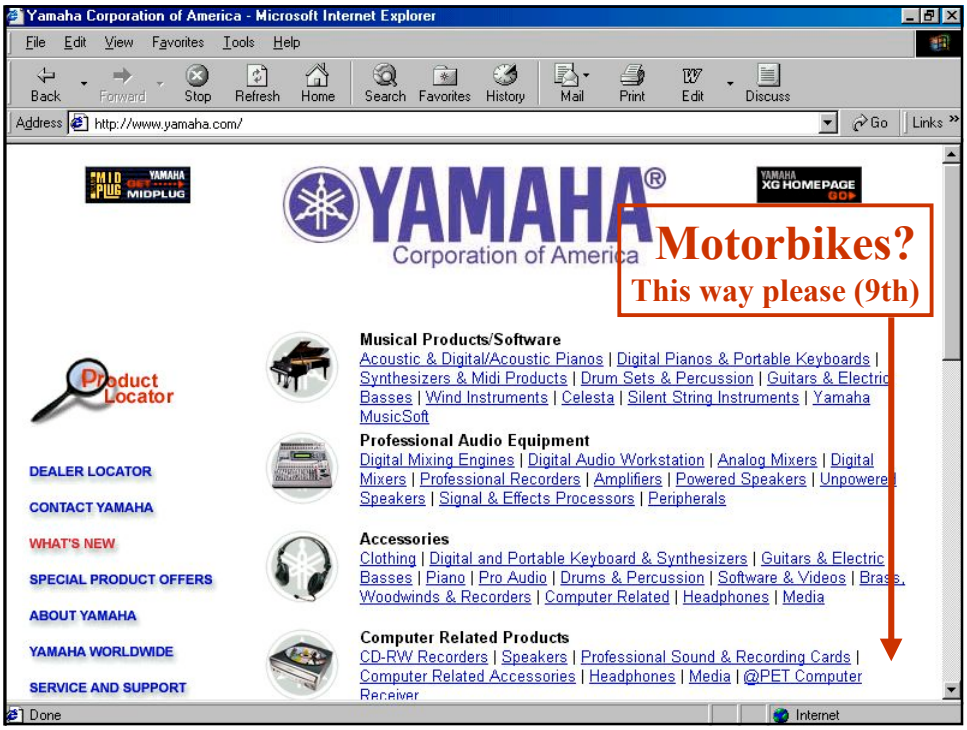
Cultural Shock : when the « has-been » meet the « superhype »



ps_testWare
Software Testing Services

Low Visibility

Hello, is anybody there?



ps_testWare
Software Testing Services

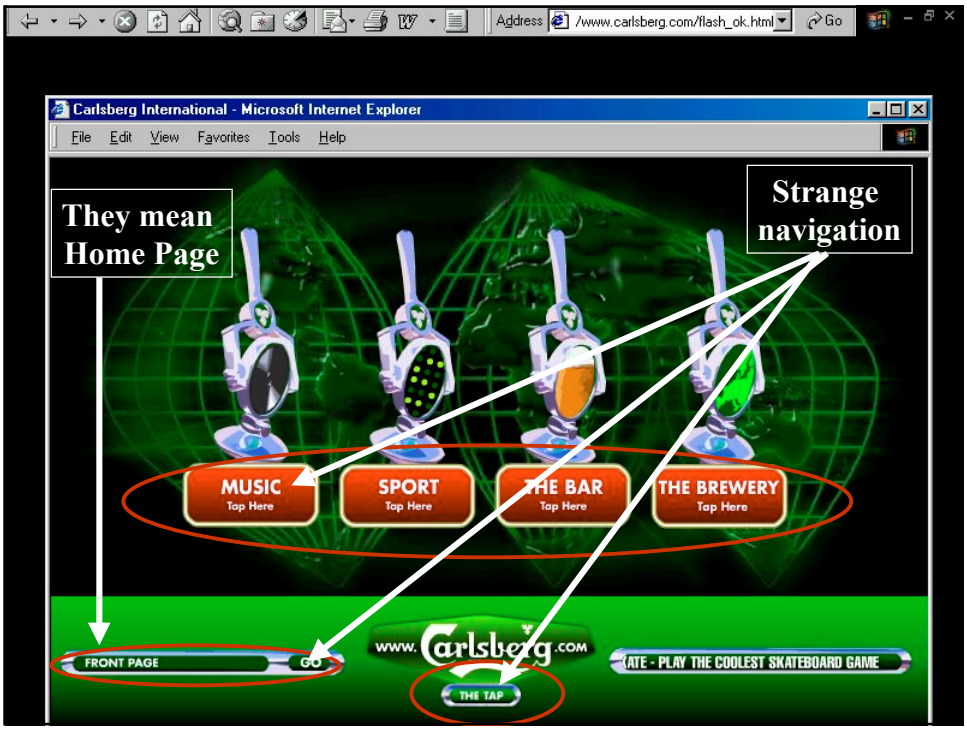
Non-Printer-Friendly Pages

Stay on-line a little bit more…I have to write it down!

ps_testWare
Software Testing Services

Unexpected Behaviour

Have I done something wrong?

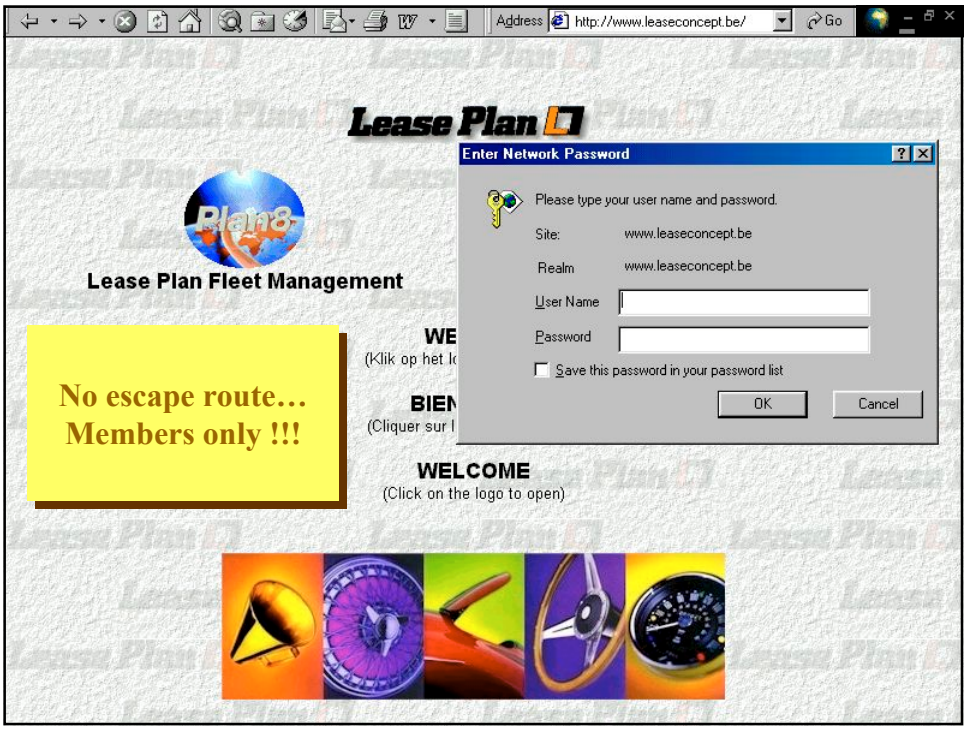


ps_testWare
Software Testing Services

Unclear Access Strategy

« Tire la chevillette et la bobinette cherra »

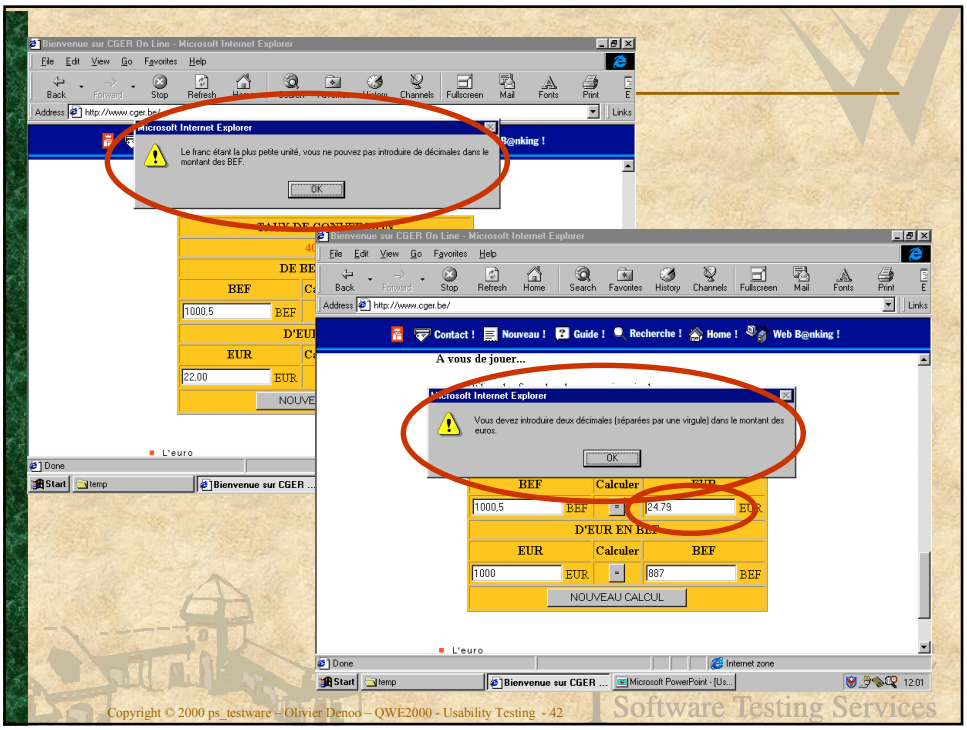
Le petit chaperon rouge



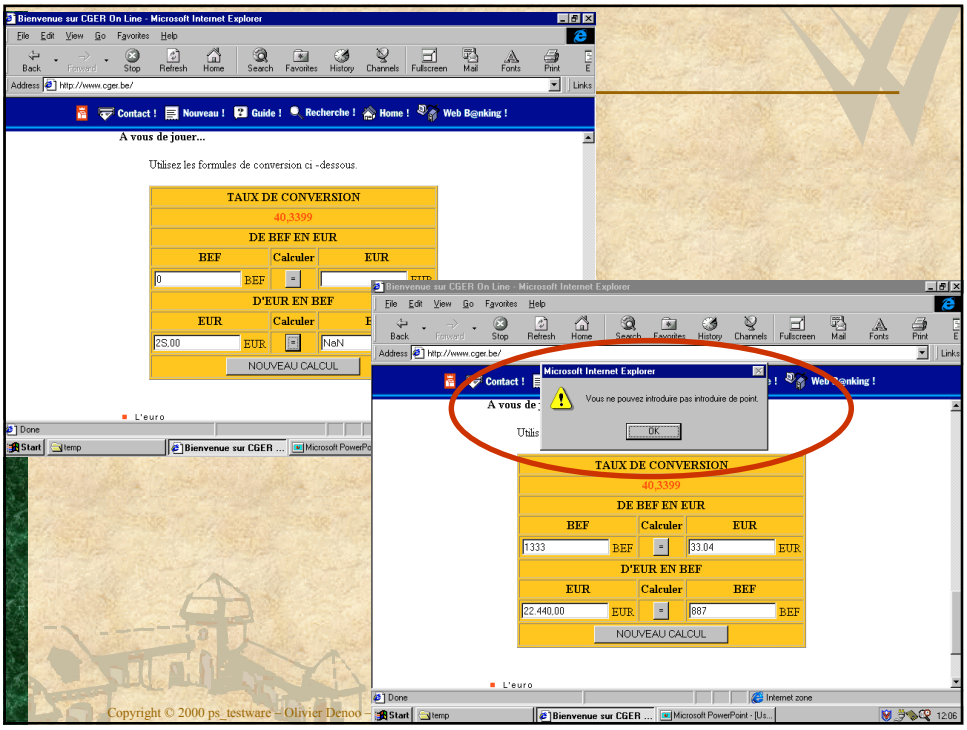
ps_testWare
Software Testing Services

Script Failures – Lack of Shielding

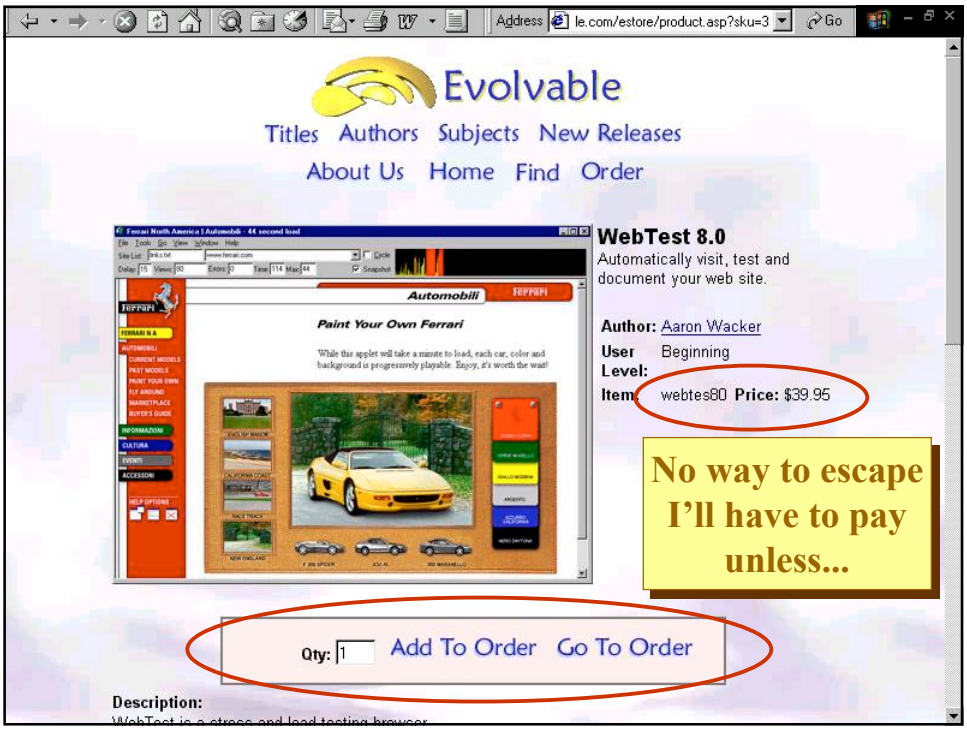
The format does not matter, but please, make your choice!

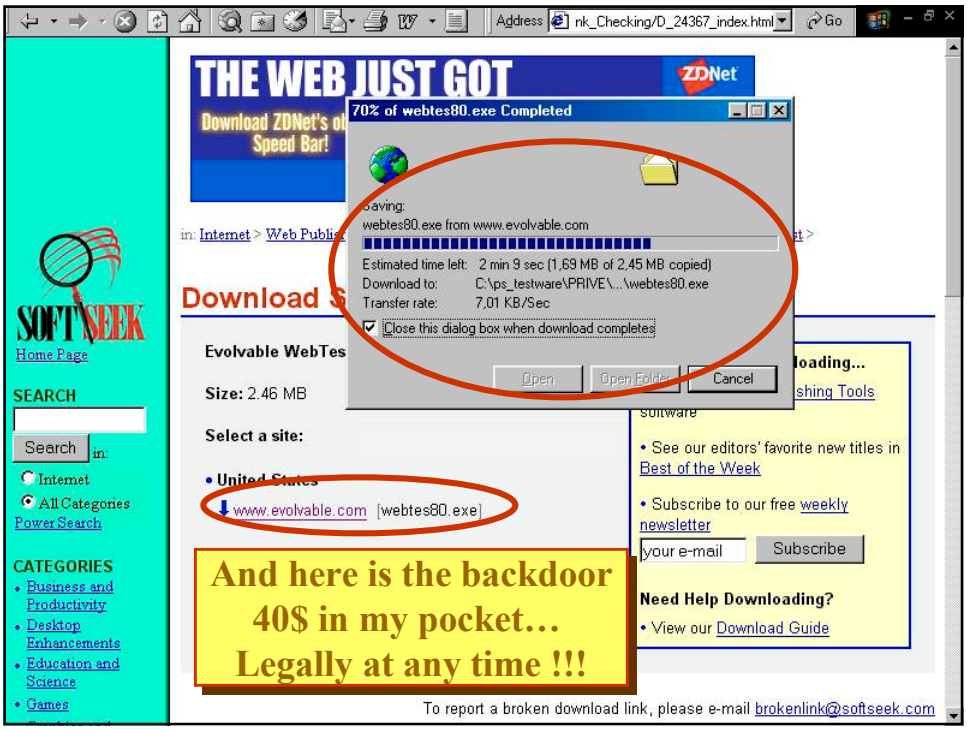


Software Testing Services


Bienvenue sur CGER On Line - Microsoft Internet Explorer
Contact !
Nouveau !
Guide !
Recherche !
Home !
Web Banking !
À vous de jouer...
Utilisez les formules de conversion ci -dessous.
TAUX DE CONVERSION
40,3399
DE BEF EN EUR
BEF
Calculer
EUR
D'EUR EN BEF
NOUVEAU CALCUL
Microsoft Internet Explorer
Vous ne pouvez introduire pas introduire de point.
OK
1333
33.04
22.440,00
887
L'euro
Copyright © 2000 ps_testware – Olivier Denoo –

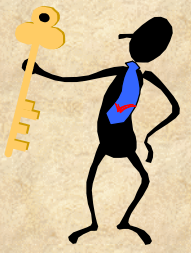

ps_testWare
Software Testing Services
Lack of Control over Global Business Strategy
Did I turn off the tap before leaving?
Copyright © 2000 ps_testware - Olivier Denoo – QWE2000 Usability Testing 44


Evolvable
Titles Authors Subjects New Releases
About Us Home Find Order
WebTest 8.0
Automatically visit, test and document your web site.
Author: Aaron Wacker
User Level: Beginning
Item: webtes80 Price: $39.95
No way to escape I'll have to pay unless...
Qty: 1 Add To Order Go To Order
Description:


THE WEB JUST GOT
70% of webtes80.exe Completed
Saving:
webtes80.exe from www.evolvable.com
Estimated time left: 2 min 9 sec (1,69 MB of 2,45 MB copied)
Download to: C:\ps_testware\PRIVE\...\webtes80.exe
Transfer rate: 7,01 KB/Sec
Close this dialog box when download completes
Open Open Folder Cancel
SOFTSEEK
Home Page
SEARCH
Search in:
Internet
All Categories
Power Search
CATEGORIES
Business and Productivity
Desktop Enhancements
Education and Science
Games
Size: 2.46 MB
Select a site:
www.evolvable.com [webtes80.exe]
• See our editors' favorite new titles in Best of the Week
• Subscribe to our free weekly newsletter
your e-mail Subscribe
Need Help Downloading?
• View our Download Guide
To report a broken download link, please e-mail brokenlink@softseek.com
And here is the backdoor 40$ in my pocket… Legally at any time !!!

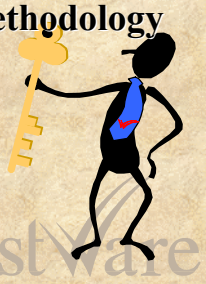
ps_testWare
Software Testing Services

Conclusions



Conclusions:

- **Issues**
  - **Maturity problem towards testing especially in new technologies**
  - **Please educate the market (www / e-commerce)**
  - **Money is not (necessarily) the issue...nor time**
- **Solutions**
  - **Structured E-Commerce Testing / Methodology**
  - **Start testing as early as possible**
  - **Business Process Thinking TM**
  - **Understand the needs and adopt the right approach (TRH)**



ps_testWare
Software Testing Services

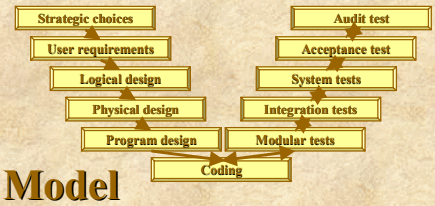
ps_testware

Software Testing Services

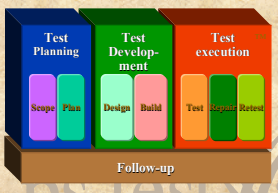
Tiensesteenweg 329
B-3010 Leuven
Tel.: +32 (16) 35.93.80
Fax: +32 (16) 35.93.88
e-mail: ps_testware@compuserve.com
http://E-Commerce.pstestware.com

Our Business

- **Structured Software Testing**
- **Methodology**



- **Implementation Model**



ps_testware
Software Testing Services

Our Services

- **Training** (see ps_testware institute)
- **Coaching**
- **Consultancy**
- **Outsourcing** (now also Total Outsourcing)

Provided by:

– Test Consultants

ps_testware
Software Testing Services

Our Products

- **Test Assessment**
- **Test Assignment**
- **Test Plan**
- **Test Report**
- **Test Advice**
- **Test Pack™** New
- **Test Laboratory** New
- ***Tools***
- ***E-commerce testing***

ps_testware
Software Testing Services

References

- Kredietbank
- Barco Graphics
- Exact Maatwerk
- ING Bank
- Bank Card Company
- Janssen Pharmaceutica
- Tessa
- Europese Raad
- Lernout & Hauspie
- Origin
- Specs
- Dexia
- Siemens
- ING 2
- Yokogawa
- Link
- Alcatel Bell
- Mobistar
- AXA-Royale Belge

Credo

ps_testware's first responsibility goes to the customers who use our services. Our services must be of high quality and must be a reference for our customers. In line with our primary business, Structured Software Testing, we may not indulge in pressure, quantity or quick profit.

We are responsible to our members, the men and women who work with us. Every member must be respected as an individual and must be rewarded personal and fair. We must support our members via a competent management, an adequate working environment and proper working conditions. Our members must have the means to provide and receive feedback, allow them and the organisation to learn continuously. We must support our members in their family responsibilities. Our actions must be just and ethical.

Our final responsibility is to our stockholders. Our business must make a sound profit. We must innovate and continuously improve our methods and techniques. We must develop new services and implement them effective and efficient. We must create reserves to provide for adverse times. Our stockholders must receive a fair return on their investments.

The Mission

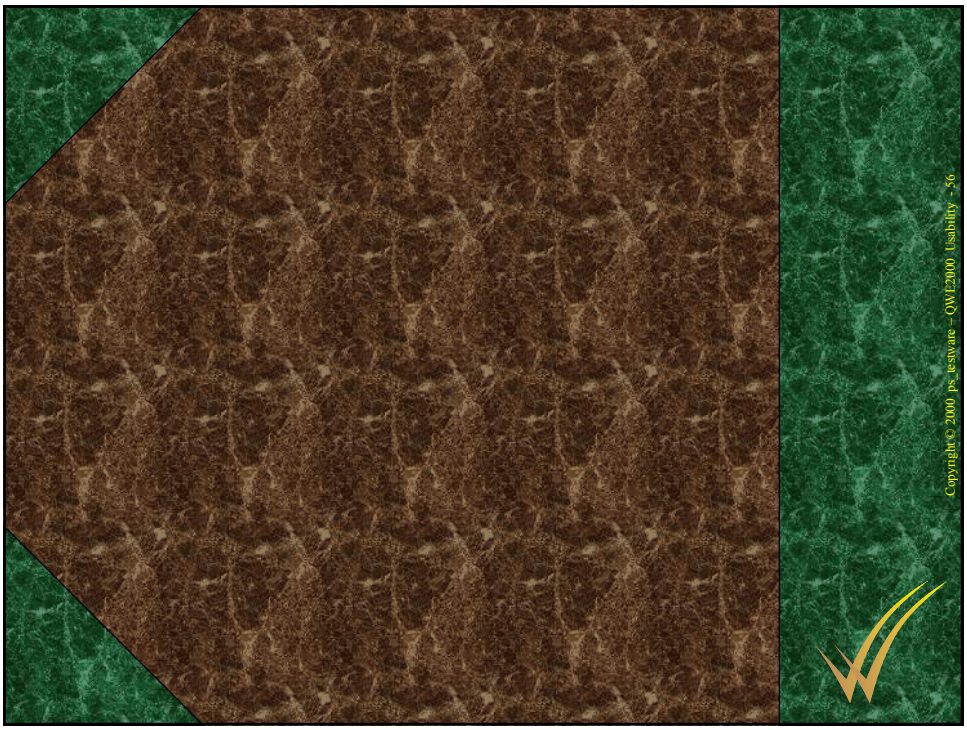
To offer the best solution to quality problems of computer systems by using its test expert knowledge in a professional way.

***Best solution**: the solution that provides the highest contribution.*

***Test expert knowledge**: the intellectual asset of ps_testware, a profound and complete knowledge regarding verification and validation (testing).*

***Professional**: the courage to really provide what has been promised.*

ps_testWare
Software Testing Services



QWE2000 Session 6M

Mr. Karl Lebsanft & Mr. Thomas Mehner [Germany]
(Siemens AG)

"CMM in Turbulent Times - Is CMM a Contradiction to Innovation?"

Key Points

- CMM - Capability maturity model of SEI.
- CMM is often misunderstood and this leads to criticism
- CMM is a good framework for improving software development processes

Presentation Abstract

CMM is applicability in improving development processes. (whether innovative or not). Concrete examples from various domains such as telecom, industrial automation, medicine, etc. will be used where return -on- investment can be shown.

About the Speaker

Karl Lebsanft studied mathematics and computer science. He is a project manager with the department 'Processes for Software and Systems' of Siemens Corporate Technology in Munich, Germany. Hi is working as a company internal consultant for software process analysis and improvement. He has nearly 20 years of experience in software engineering, project management, and process innovation. He has led the execution of numerous CMM-based maturity level assessments worldwide. He coaches the introduction of measurement programs on all organizational levels, from development tup to the management and business levels. In addition, his responsibilities include research on measurement and evaluation of projects and processes. His research interests are located in the fields of process modeling, simulation, metrics, and project estimation.

Thomas Mehner received his diploma in computer science from the Technical University of Munich in 1982. In 1983 he joined the corporate research and development division of the Siemens AG. He is a project manager in the department 'Processes for Software and Systems' of Siemens Corporate Technology in Munich, Germany. With seven years of experience as a company internal process consultant, he is on of the most-experienced lead assessors in this area at Siemens Corporate Technology in Munich, Germany. With seven years of experience as a company internal process consultant, he is one of the most-experienced lead assessors in this area at Siemens. He is particularly responsible for the area of process innovation (e.g. incremental processes, spiral development, make or buy decisions, how

component-oriented development influences the development processes, etc.). He has particular domain expertise in areas of telecommunications, industrial automation, and defense.

S ZT SE 3

CMM in Turbulent Times – does CMM Contradict Innovation and Flexibility ?

- **CMM and Assessment-based Improvement**
- **CMM: Strengths and Limitations**
- **Summary**

Karl Lebsanft,
Siemens AG Phone ++49-89-636-53365
ZT SE 3 Fax ++49-89-636-44424
Karl.Lebsanft@mchp.siemens.de

Thomas Mehner
- 47253
Thomas.Mehner@mchp.siemens.de

S ZT SE 3

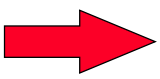
Significance of Software within Siemens

Software is of strategic importance for many business fields

- ☐ Most products include a significant proportion of software
- ☐ Costs for development, maintenance and service are in the billions (DM)

Software development conditions are becoming more demanding

- ☐ Costs and deadline pressure are increasing
- ☐ Quality and flexibility requirements are increasing
- ☐ Systems and development processes are becoming continually more complex
- ☐ Distributed development between many sites / countries



Quality and efficiency are increasingly becoming competition factors in system and software development

S ZT SE 3

Siemens Software Initiative

Keeping software expertise at Siemens among the best worldwide

Challenge

Business success through world-class software and engineering competence

Goals

Business-oriented Improvement with measurable results

Benchmarking & Strategies for software-related business (software, systems, and engineering of industrial plants)

Experience Sharing & Best Practice Networking, Just-in-time Training

Enterprise-wide (de facto) Standards:

- Siemens Assessments,
- Siemens "top^{Six}" balanced scorecards,

Benefits

Focus on Customer Benefit

Controlling Process- and Product-Quality

Reaching Business Goals

Cooperative Product Strategies

Siemens-wide Software Community, Learning Organizations

S ZT SE 3

Siemens Process Assessments

Siemens Process Assessments are cornerstones in Siemens' strategic Program to increase performance in software development.

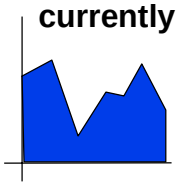
They are used in many business fields to :

- Analyze and assess processes and architectures
- Determine improvement potential
- Specify measures to improve processes
- Implement programs to increase effectiveness

S ZT SE 3

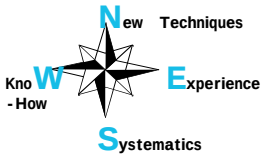
Assessments clarify Issues of Key Importance to the Organization

- **What software development methods are currently practiced within the organization?**



- **How do we do compared with the competition and with other Groups?**
- **Where should we bundle our resources to make the most improvements?**

Main outcome:
What provides the max. leverage despite resource restrictions?



R I SK MANAGEMENT
M ETRICS
P ROJECT CONTROLLING
DIST R IBUTED DEVELOPMENT
PR O JECT MANAGEMENT
RE V IEWS
INT E RIM PROFILE
SUB-CONTRACTOR M ANAGEMENT
INCR E MENTAL PROCESS
CO N FIGURATION MANAGEMENT
QUALI T Y ASSURANCE
u.s.w.

S ZT SE 3

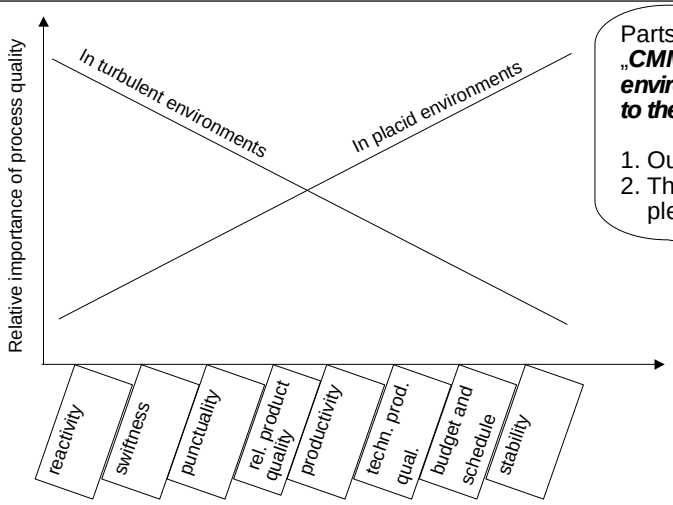
What do we expect from a process ?

- ☐ Organize work to ensure business success.
- ☐ Everyone should be aware about the duties he is in charge of.
- ☐ Everyone should be clear about the results he can rely on.

CMM always provides invaluable hints on how to come up with the adequate solution.

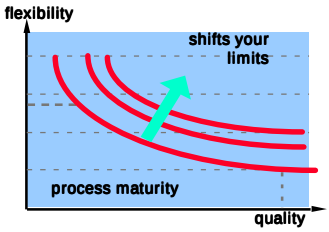
S ZT SE 3

Turbulent vs. placid environments



Parts of the SW community claim:
„CMM fits for pleasant and stable environments, but it is a contradiction to the turbulent world“

1. Our experience: That is not true!
2. There are only very few pleasant environments



CMM in Turbulent Times - QW2000

S ZT SE 3

Capability Maturity Model (SEI)

Maturity Level	Characteristics	Benefits
5 “Optimizing”	• Management of changes to the Process • Error prevention process • Management of technological changes	Quality
4 “Managed”	• Quantitative goals for Product and Process • Statistical Control of goal attainment • Reuse	
3 “Defined”	• Process Owner is the Organization • Organization-wide Standard Process • Projects derive their process from Stand. Proc.	
2 “Repeatable”	• Process “owner” is Project Manager • Strict Project Management • Process varies from Project to Project	
1 “Initial”	• Process undefined; Ad-Hoc methods • Only a few individual experts make decisions • Deadlines, quality & costs cannot be predicted	Risk

CMM in Turbulent Times - QW2000

S ZT SE 3

Business goals guide process innovation

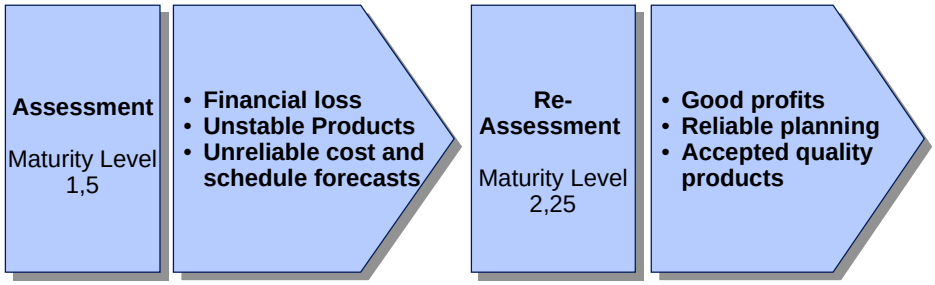
Only basic process in place:
Spread personal process know-how
Unification of procedures
Repeatable project success
→ **Process Documentation!**

Established process:
Optimization of specific aspects
Introduction of new methods
and procedures
→ **Process Optimization!**

Business changes
Markets are changing
→ **Process (re)engineering!**

S ZT SE 3

Example from one of our Groups



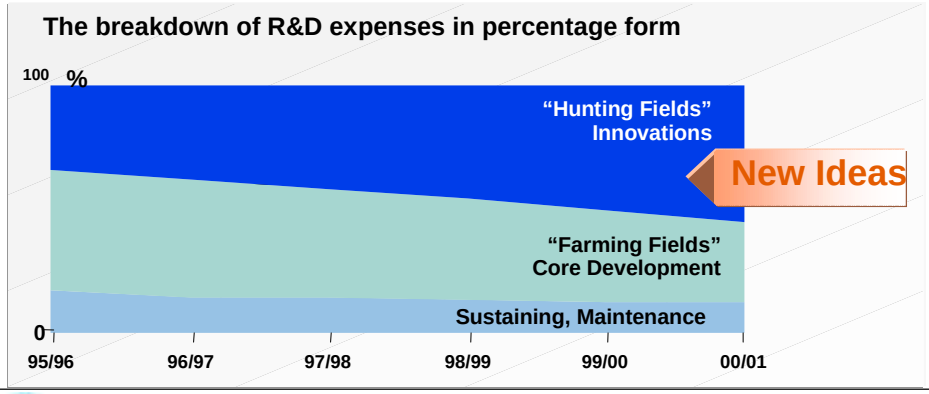
In parallel for two trial projects "creative procedures without process rules"=>
- **Cost and schedule overrun (> 100%)**
- **Results: No product (fulfilling market expectations)**

S ZT SE 3

Improvement Creates Room for Innovation
Best Practice Example

Attaining high maturity (Process Capability) leads to:

- less effort for current products and their maintenance
- more room for innovation; new products and markets



CMM in Turbulent Times - QW2000

S ZT SE 3

The Scope of CMM has limitations:
CMM alone is not enough – you need good Products

Success comes from:

Doing the right things right.

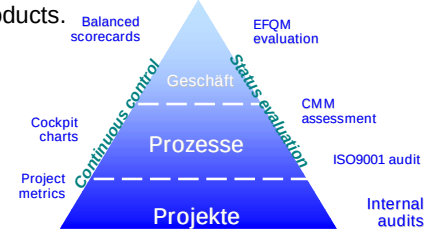
Not covered by CMM

"The right things":

- Scanning and selection of business opportunities.
- Definition of competitive products.

CMM Focus

"Do it right":
Define and live an adequate development process.



CMM in Turbulent Times - QW2000

S ZT SE 3

The Scope of CMM has limitations: CMM is not enough – you need the best people

Success comes from:

The company has the best people with excellent skills, working in fruitful teams

Not covered by CMM → Selection and strategic development of staff and management

CMM Focus → Training and inter working aspects

S ZT SE 3

CMM tells you what to take care of – not how to do it

Success means:

Establish adequate (market, product, size,...) procedures for all relevant activities.

Not covered by CMM → Which methods, necessary flexibility, tooling,...

CMM Focus → Areas and topics which have to be under control:

- Craftsmanship (Level 2)
- Engineering (Level 3 and 4)

S ZT SE 3

CMM does not require huge process documentation

Success means:

Process documentation for efficient support of developers

Not required by CMM

Lots of paper, one process for all

CMM Focus

Document sufficiently to make good results repeatable

Suitable availability and the ability to check whether the process and associated documentation are being adhered to

Continuous learning

Systematic Tailoring

CMM in Turbulent Times - QW2000

S ZT SE 3

CMM does not require a specific process model

Success means:

Market-oriented development processes

Incremental development requires higher maturity (at least in level 2 areas) than waterfall:
- Requirements
- Configuration Management
- Project Management
- Supplier Management
- Quality Assurance

Not defined by CMM

Sequential (waterfall) or incremental or...

CMM Focus

Abilities in SW engineering and related areas:

Analysis, design, implementation, test,...

CMM in Turbulent Times - QW2000

Conclusion

- CMM is an excellent basis for success in the SW business.
- CMM is applicable in turbulent as well as in placid environments.
- CMM is no guarantee for success.
- Application of the CMM requires consideration of the environment.

QWE2000 Vendor Technical Presentation VT6

Panos Ntourntoufis (UPSPRING Software, Inc.)

"An Automated Approach to Software Defect Prevention"

Key Points

- Point 1...
- Point 2...
- Point 3...

Presentation Abstract

A solution for dramatically reducing the number of defects in large C, C++ or Java applications is described. This solution has been implemented using a software Development Information System (DIS), and consists of two stages: 1) Existing Defect Identification identifies and fixes existing software defects early in the development cycle; 2) New Defect Prevention prevents new defects from entering the code base by enforcing coding standards through automated code reviews, and preventing code breakages by enabling accurate impact analysis.

About the Speaker

Dr Panos Ntourntoufis, Ph. D., D.I.C. - Greek National. Born and educated in Belgium. Degree in Mechanical and Electrical Engineering from the Free University of Brussels, Belgium (1988). Joined the Neural Systems Engineering Laboratory at Imperial College London (1988-1992). Ph.D. in Electrical Engineering from Imperial College London, University of London (1994). Appointed Research Fellow at Brunel University, England, to carry out Neural Systems research (1994-1995). C/C++ software development for Siemens-Plessey at Eurocontrol Maastricht, The Netherlands (1996). Joined Equifax Europe, in April 1996, to carry out R&D in the field of Hybrid Risk Assessment Systems. Was co-running the Equifax Europe New Technology Forum. Joined Software Emancipation Technology (1997) where he is currently a technical manager for Europe. Has collaborated with many large software development organizations, in the Telecommunications, Manufacturing and ISV sectors, to help them achieve greater levels of Productivity and Software Quality. Speaks French, English and Greek fluently.

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

Dr Panos Ntourntoufis
European Technical Manager
Software Emancipation Technology

agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

agenda

QWE 2000 – 23rd November 2000

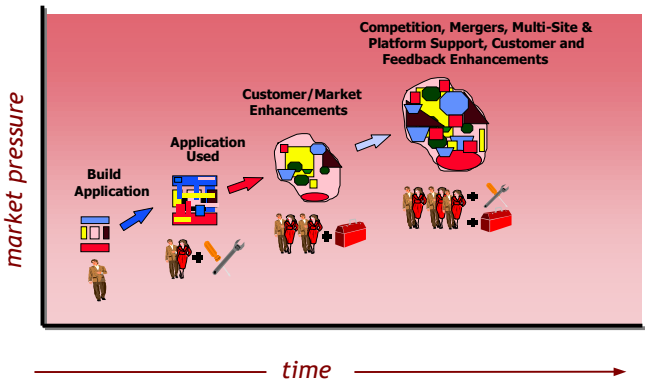
- **Software Development Crisis**
- Development Information Systems
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

development crisis

SYSTEM EVOLUTION



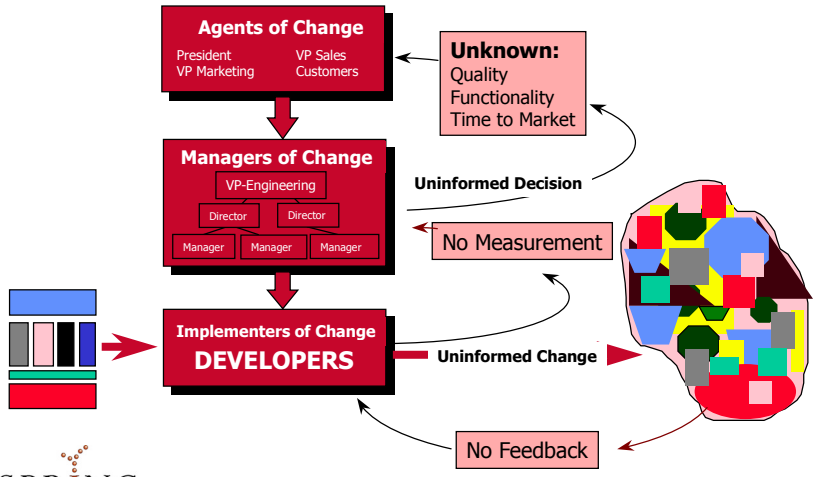
Increased Product Complexity due to market pressure creates a "Software Complexity Crisis"

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

development crisis

MANAGING CHANGE

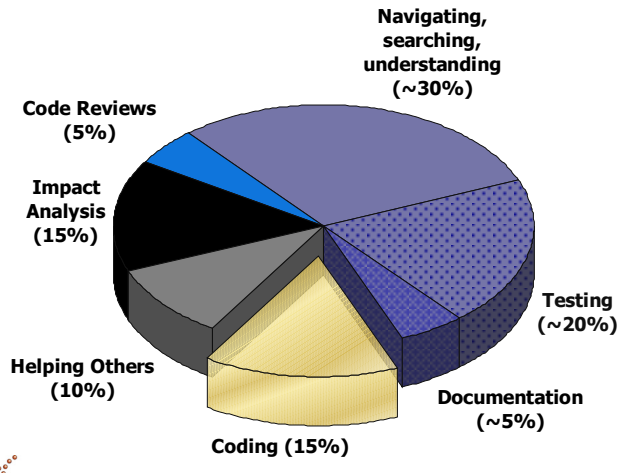


UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

development crisis

COMPETITIVE ADVANTAGE



UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

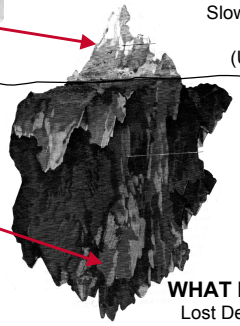
development crisis

SOFTWARE QUALITY

2 Types of Quality

External Quality
That Which Can be Seen by Customers
Traditionally Tested

Internal Quality
Program Structure
Coding Practices
Maintainability
Domain Expertise



WHAT YOU SEE
System Crashes
Unexpected Behavior
Data Corruption
Slow Performance
(Usability)
(Usefulness)

WHAT IS HAPPENING
Lost Development Time
Loss of Competitiveness and Reputation
Fixes Introduce New Problems and Lengthy Retesting
Loss of Domain Expertise Introduces Delays

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

development crisis

SOFTWARE QUALITY

- **Assertions!**
  - External Quality is a Symptom
  - Internal Quality is the Root Problem
  - Poor Internal Quality = High Maintenance Cost
  - In Order to Improve SW Quality:
    - Internal Quality Must be Improved

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- **Development Information Systems**
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

DIS

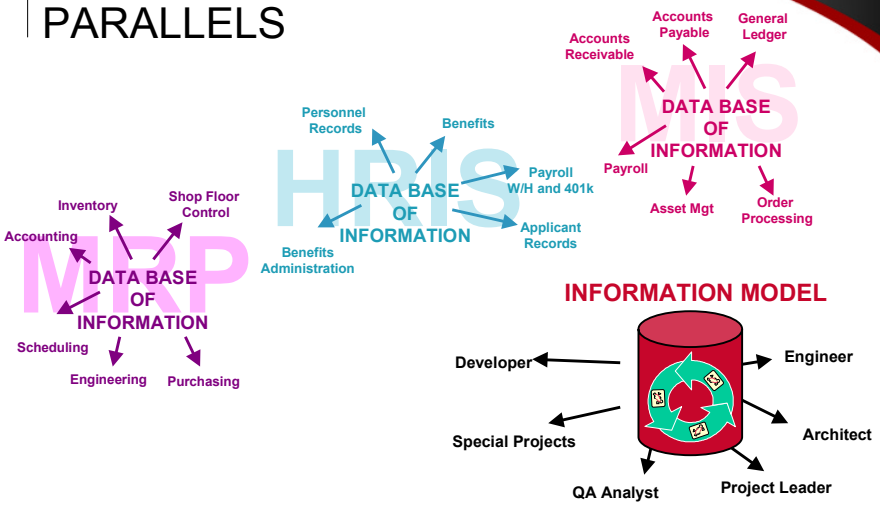
DEFINITION

A Software Development Information System (DIS) based on a **single, comprehensive database** of all the objects used throughout the software design, implementation, test, and maintenance process

The *DIS's* applications enable **analysis and modification** of the data while monitoring and controlling the processes that produce them

DIS

PARALLELS

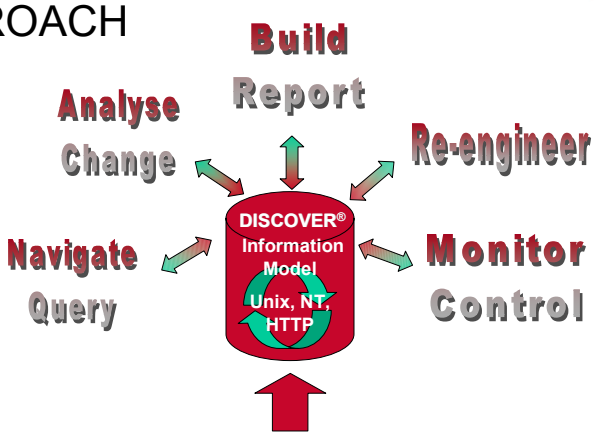


UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

DIS

APPROACH



C, C++, SQL, Java Source Code

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- **Solution Stage 1: Quality Assessment**
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

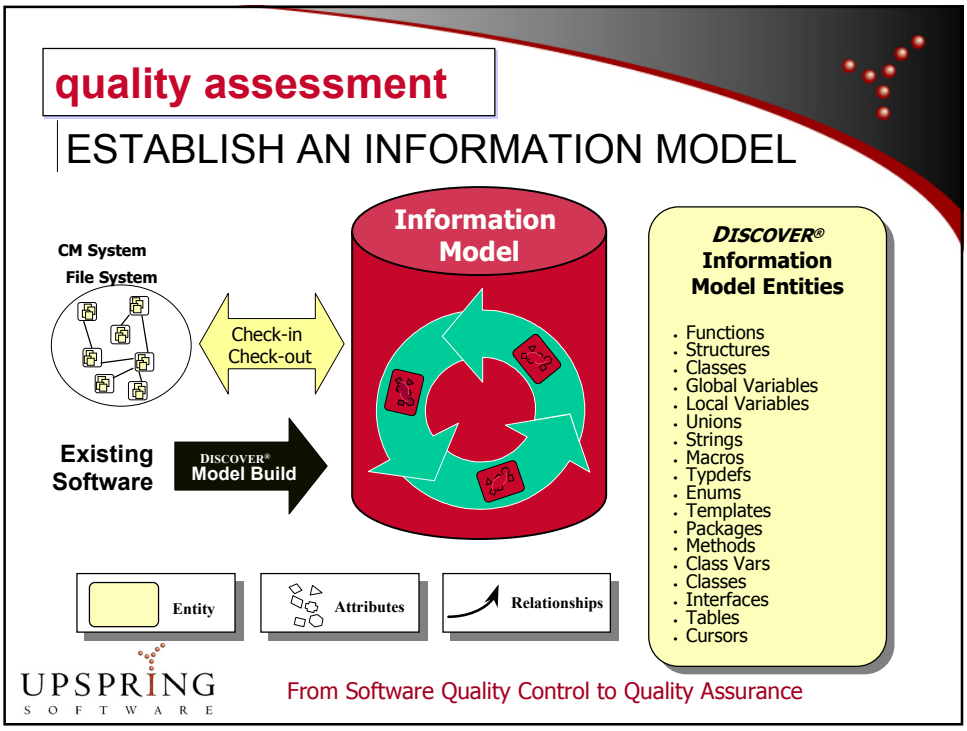
agenda

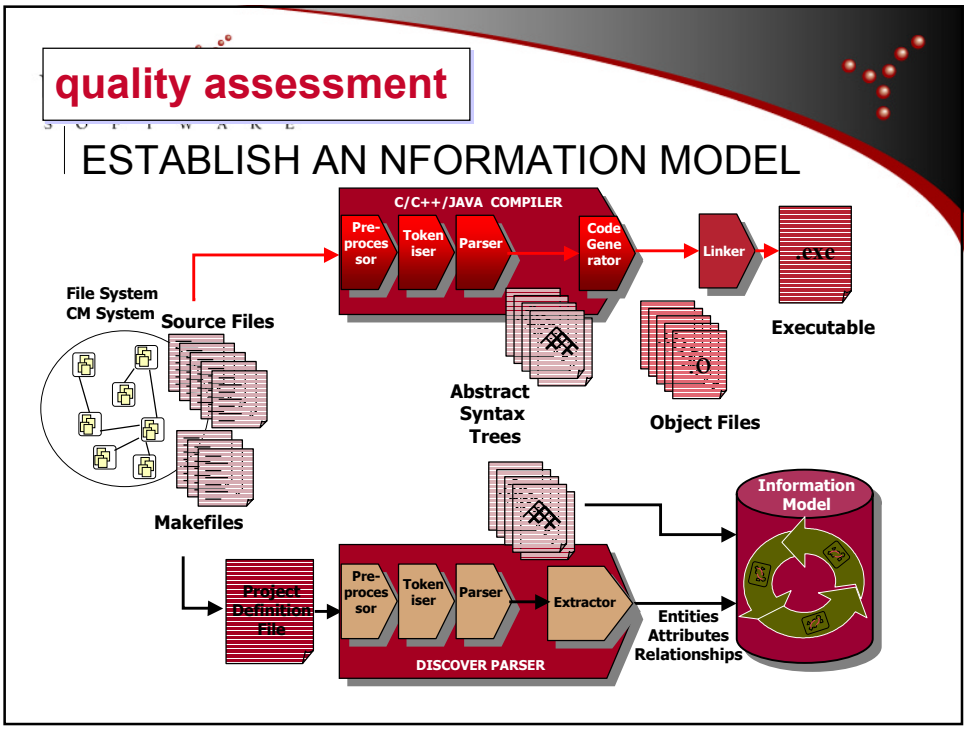
QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- **Solution Stage 1: Quality Assessment**
  - **Establish an Information Model**
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

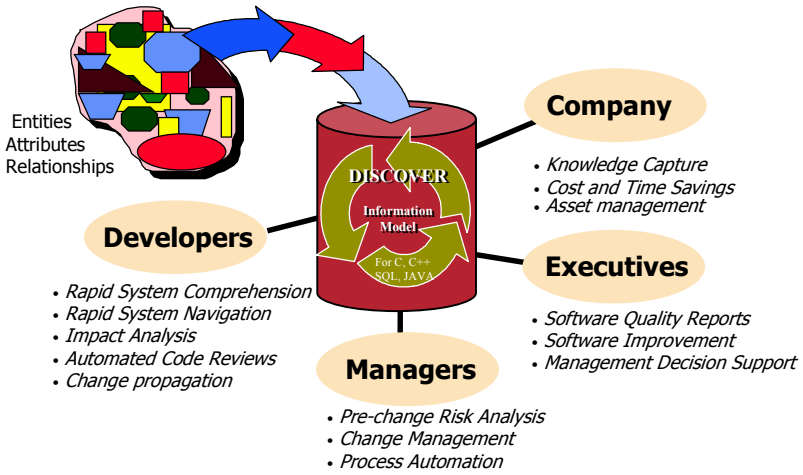

quality assessment
ESTABLISH AN INFORMATION MODEL
CM System
File System
Check-in
Check-out
Information Model
Existing Software
DISCOVER® Model Build
DISCOVER® Information Model Entities
Functions
Structures
Classes
Global Variables
Local Variables
Unions
Strings
Macros
Typdefs
Enums
Templates
Packages
Methods
Class Vars
Classes
Interfaces
Tables
Cursors
Entity
Attributes
Relationships
UPSPRING SOFTWARE
From Software Quality Control to Quality Assurance


quality assessment
ESTABLISH AN NFORMATION MODEL
C/C++/JAVA COMPILER
Pre-processor
Token iser
Parser
Code Generator
Linker
.exe
Executable
File System
CM System
Source Files
Abstract Syntax Trees
.O
Object Files
Makefiles
Information Model
Project Definition File
Pre-processor
Token iser
Parser
Extractor
DISCOVER PARSER
Entities
Attributes
Relationships

quality assessment

ESTABLISH AN NFORMATION MODEL

***DISCOVER®* provides a persistent store of information about your application unobtainable *anywhere* else:**



agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- **Solution Stage 1: Quality Assessment**
  - Establish an Information Model
  - **Establish Quality Standards**
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

quality assessment

ESTABLISH QUALITY STANDARDS

Standard Quality Filter Sets (QFS)

Over 150 Quality Standards for Software Development

Customise & Extend

quality assessment

ESTABLISH QUALITY STANDARDS

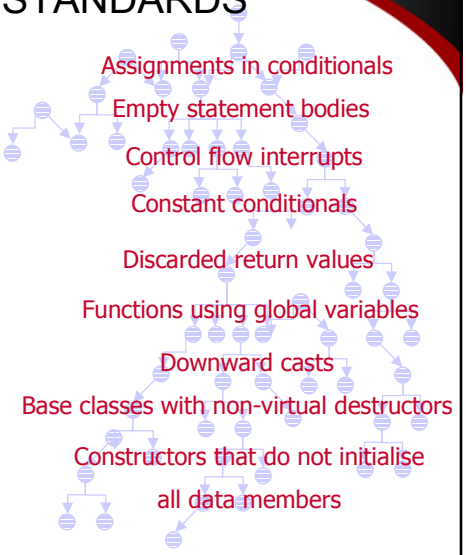
TREE PATTERN MATCHING (TPM)

Search by syntactic patterns in the abstract syntax tree

Uncover hidden suspect code

Automatically remediates code through change propagation.

highly complex and error-prone constructions can be detected and replaced before they surface later in the development process



quality assessment

ESTABLISH QUALITY STANDARDS

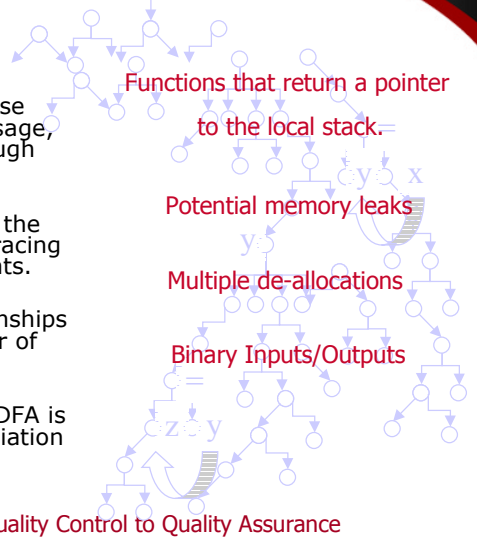
DATA FLOW ANALYSIS (DFA)

Data use is often masked because variable names are unrelated to usage, or because data are passed through functions

DFA finds all entities that receive the value of a variable by recursively tracing through all assignment statements.

"in", "out" or "bi-directional" relationships are explored through any number of levels

The accuracy and completeness of DFA is critical for bug fixes or other remediation efforts.



UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

agenda

QWE 2000 – 23rd November 2000

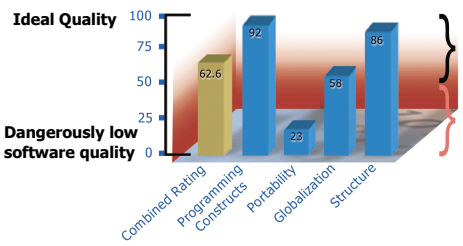
- Software Development Crisis
- Development Information Systems
- **Solution Stage 1: Quality Assessment**
  - Establish an Information Model
  - Establish Quality Standards
  - **Assess initial quality and identify defects**
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

quality assessment

ASSESS QUALITY AND IDENTIFY DEFECTS



Low risk/ high quality

Products to market sooner, fewer defects

High risk/low quality

Products to market later, more defects

- **Assess Initial Quality and Identify P1 Defects**
  - Over 150 quality rules used (customisable)
- **Quality Filter Based QAR (Quality Assessment Report)**
  - Graphical charts for management
  - List of viewable suspect instances for engineers
  - Quantitative baseline to measure improvements in product quality

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- **Solution Stage 1: Quality Assessment**
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - **Repair defects**
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

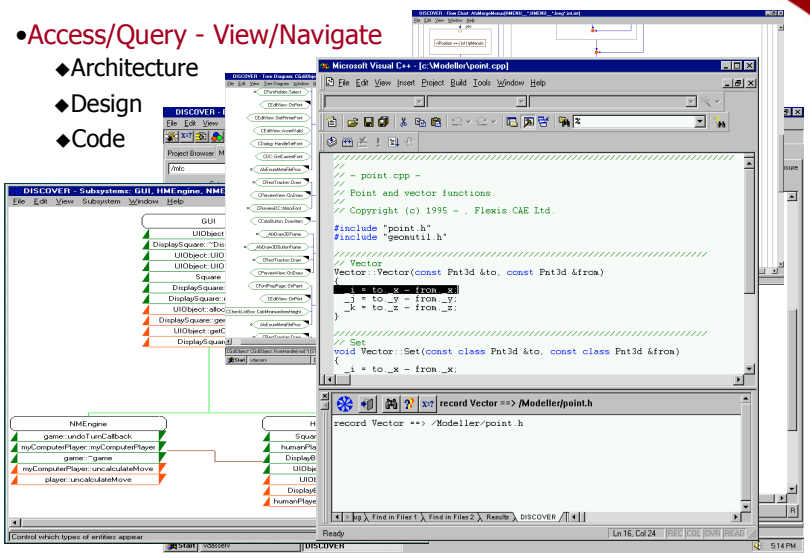
UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

quality assessment

REPAIR DEFECTS

- Access/Query - View/Navigate
  - Architecture
  - Design
  - Code



agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- **Solution Stage 1: Quality Assessment**
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - **Monitor Quality**
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

UPSPRING SOFTWARE

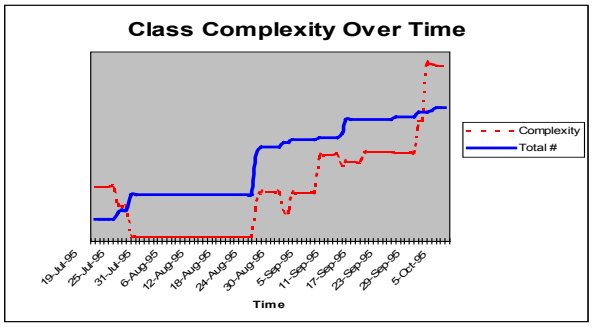
From Software Quality Control to Quality Assurance

quality assessment

MONITOR QUALITY

Trend Analysis

- Regularly Generated Quality Assessment Reports
- Provides Assessment of Trends Over Time



agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- **Solution Stage 2: Quality Assurance**
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- **Solution Stage 2: Quality Assurance**
  - **Impact Analysis**
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

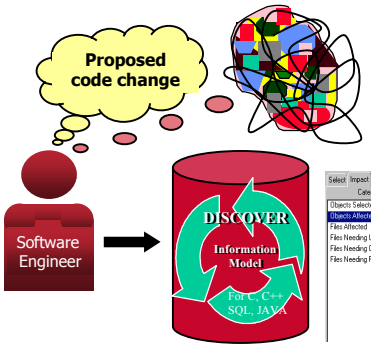
UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

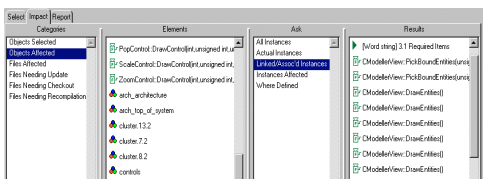
quality assurance

IMPACT ANALYSIS

- Examines the impact of proposed entity signature changes
- ***before*** they become final
- Prevents further defects from entering the code
- Reliable, accurate, and immediate



- *How many places will be affected?*
- *How much effort is required?*
- *What are the recompilation dependencies?*
- *Will the impact be local or will it cross project boundaries?*



agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- **Solution Stage 2: Quality Assurance**
  - Impact Analysis
  - **Prevent defects – check Config. Mgmt. Submissions**
  - Application simplification

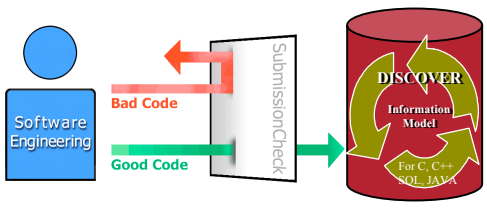
UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

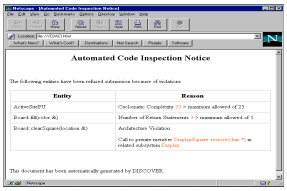
quality assurance

PREVENTS DEFECTS – CHECK CM SUBMISSIONS

- **Providing a means to prevent defects early**
  - Monitors compliance of new or modified code with established quality standards via QFS ***before*** submission
  - Early detection of violations in coding procedures
  - Allows process repeatability & scalability



Compliance Report



UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- **Solution Stage 2: Quality Assurance**
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - **Application simplification**

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

quality assurance

APPLICATION OPTIMISATION

- **Elimination of Unnecessary Header Files**
  - reduce header file inclusions to only those necessary
    - Reducing footprint
    - Reducing compile time
    - Simplifying Maintenance
- **Elimination of Dead Code**
  - identify code that is not used by a user-specified root entity
  - only live code for dev teams to maintain, test, or remediate
- **Other Areas**
  - Reorganising software structure & functionality
  - Porting between C, C++, Java
  - Porting to new platforms
  - Internationalising software

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

agenda

QWE 2000 – 23rd November 2000

- Software Development Crisis
- Development Information Systems
- Solution Stage 1: Quality Assessment
  - Establish an Information Model
  - Establish Quality Standards
  - Assess initial quality and identify defects
  - Repair defects
  - Monitor Quality
- Solution Stage 2: Quality Assurance
  - Impact Analysis
  - Prevent defects – check Config. Mgmt. Submissions
  - Application simplification

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

agenda

QWE 2000 – 23rd November 2000

UPSPRING SOFTWARE

http://www.upspringsoftware.com

UPSPRING SOFTWARE

From Software Quality Control to Quality Assurance

An automated approach to software defect prevention

Panos Ntourntoufis
Software Emancipation Technology Ltd.
Reading, United Kingdom
panos@setech.com

Abstract

A solution for dramatically reducing the number of defects in large C, C++ or Java applications is described. This solution has been implemented using a software Development Information System (DIS), and consists of two stages: 1) *Existing Defect Identification* identifies and fixes existing software defects early in the development cycle; 2) *New Defect Prevention* prevents new defects from entering the code base by enforcing coding standards through automated code reviews, and preventing code breakages by enabling accurate impact analysis.

1 Introduction

Improving quality, increasing productivity, reducing software complexity, and introducing controlled software measurement, remain the aims of a large number of software development initiatives [1][6]. The financial justification for improving process is obvious - software development pays a extremely high price without optimal processes in place. As software ages and complexity increases due to market pressure, maintenance costs increase exponentially whilst product quality steadily decreases. According to the U.S. Defense Dept. and the Software Engineering Institute at Carnegie Mellon University, there are typically 5 to 15 flaws in every 1,000 lines of code. Just tracking down each defect takes about 75 minutes, according to a five-year Pentagon study. And fixing them takes two to nine hours each. On the outside, that is 150 hours, or roughly $30,000, to cleanse every 1,000 lines of code [5].

The following sections of the paper describe a solution for dramatically reducing the number of defects in the code base. This solution has been implemented using information built in a Software Development Information System [7][8], and consists of two stages:

Stage 1: Existing Defect Identification aims at identifying and fixing software defects before the customer finds them.

Stage 2: New Defect Prevention aims at preventing new defects from entering the code base by enforcing coding standards through automated code reviews, and preventing code breakages by enabling accurate impact analysis.

2 Software Development Information Systems

A software *Development Information System* (DIS) is defined as a database that captures the interrelationships between all entities in the code base (files, functions, macros, variables, etc), and its associated tools - to provide the critical information needed both by management and development teams, in order to increase productivity, quality and process [Figure 1]. The information that populates the database, also referred to as *Information Model*, is typically generated by compiler-level parsing of the source code.

The concept of Information System has been applied to other industries for a number of years and has produced systems used in manufacturing (such as MRP and ERP), in human resources (HRIS), and design (CAD/CAM) [2]. It is only recently that the idea of a DIS aimed at Software Development teams has emerged. For a review of the subject we recommend papers by Rogers [1], Boes [2] and GartnerGroup [3].

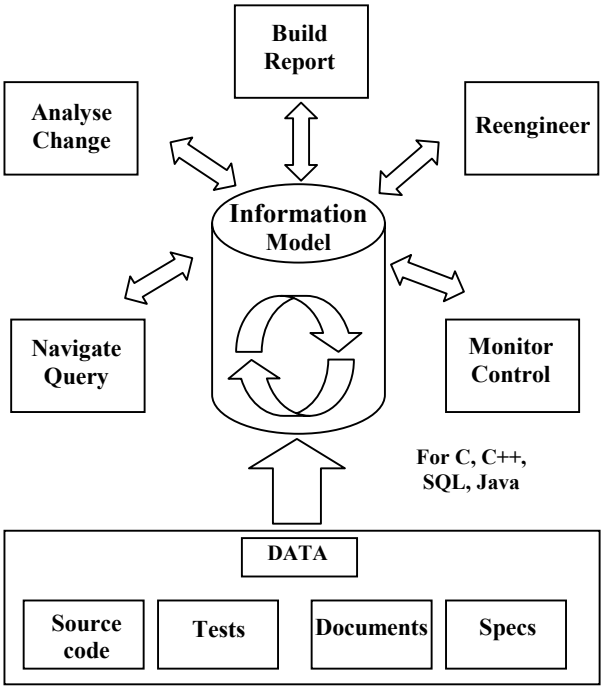


Figure 1. Software Development Information System.

3 The Software Development Crisis

3.1 Definition

As software ages and complexity increases due to market pressure, maintenance costs increase exponentially whilst product quality steadily decreases. As applications mature, source code grows in size and complexity, resulting in poor quality and long release cycles. In addition, software development consists of multiple error-prone tasks such as navigation, impact analysis, code review and comprehension that are tedious, difficult to perform, and often do not yield complete or accurate results. The increased product complexity due to market pressure creates a "Software complexity crisis".

3.2 External and Internal Software Quality

Two types of software quality can be distinguished: *External Quality* and *Internal Quality*.

External quality is that which can be seen by the customers and which is traditionally tested. Bad external quality is what can be seen in system crashes, unexpected behavior, data corruption or slow performance.

Internal quality is the hidden part of the iceberg, i.e. program structure, coding practices, maintainability, domain expertise. Bad internal quality will result in lost development time, fixes are likely to introduce new problems and therefore require lengthy re-testing. From a business point of view, this will invariably result in loss of competitiveness and reputation.

We argue that external quality is a symptom whereas the root problem is internal quality. Poor internal quality leads to high maintenance costs. In order to improve software quality, internal quality must be improved.

4 Stage 1: Existing Defect Identification

4.1 Introduction

A software defect and repair cost 10 to 100 times as much to fix during testing phase if not caught in earlier in design and coding. Studies have also demonstrated their cost will grow up to 40 to 1000 times the original if not found until after software has been released [4]. Finding defects earlier in the cycle saves therefore over $10,000 per defect.

The following sub-sections describe a series of steps which will enable to identify and fix defects before the software is released to the customer.

4.2 Establish an information model

Establishing an information model requires the following steps:

1- build and deliver the initial information model, the database that captures all software entities, their attributes and inter-relationships, together with software artifacts such as tests and documentation.
2- synchronise the model build process with the software build process itself
3- provide training to the administrators who will maintain the information model
4- ensure the repeatability of the model build process

4.3 Establish standards for software development: standard quality filter sets (QFS)

The DIS provides a significant number of industry-standard *quality filter sets (QFS)*, each covering a different category of software quality standards. Typical QFSs include *programming constructs*, *software structure*, *adherence to globalisation standards*, *portability*, *statistics*, *metrics*, etc.

As an example of QFS, we consider *Programming Constructs* queries, which measure the use of questionable programming constructs that may be unsafe or may conflict with site-specific programming guidelines. Use of questionable programming constructs adds to overall complexity of the code base, compromises security and makes the code less manageable and less modular than intended. Examples of questionable programming constructs include *empty statement bodies, functions using global data, functions that return a pointer to a local stack, potential memory leaks,* etc.

The creation of QFSs is enabled by two unique technologies built in the DIS: Tree Pattern Matching (TPM) and Data Flow Analysis (DFA).

TPM is a language that allows the navigation of the Abstract Syntax Tree (AST) generated by the DIS when the Information Model is built. Quality queries are therefore constructed by translating a particular code construct into an AST pattern. Identifying all the places in the code that match a particular query is then accomplished by finding all the places in the AST that match the corresponding pattern.

The Data Flow Analysis (DFA) functionality is also based on the Tree-Pattern Matching (TPM) technology. DFA searches for all entities that receive the value of a variable by tracing through all the assignment statements. DFA enables the identification of instances of memory allocations with no corresponding de-allocations and this globally throughout the application.

The TPM and DFA technologies can also be used to create company-specific quality filters.

4.4 Assess initial quality and identify defects

The quality of the source code is assessed by applying the QFSs previously defined and collecting results on the number of instances detected. Quality assessment is then represented graphically using a chart of

quality indexes, each index corresponding to a QFS. The quality index QI_F for the quality filter set F is calculated using the expression

$$QI_F = 100 \cdot \left(1 - \sum_q w_q \cdot \left\lceil \frac{3 \cdot (\text{nb. defects found})_q}{(\text{total relevant lines of code})_q} \right\rceil \right),$$

where $\lceil \ \rceil$ is the *ceiling-to-1* function, w_q is a normalised *weight* associated with query q, with $\sum_q w_q = 1$, and the scaling constant 3 expresses the heuristic that if a third of the code examined is severely error-ridden, it probably needs a significant amount of remediation and therefore the quality index for the correspondent query is given the value 0. For instance, Figure 2 shows the overall quality index together with the quality indexes of four quality filter sets (QFS): programming constructs, portability, globalisation and structure. Quality indexes approaching 100 are representative of near ideal quality whereas low quality indexes are representative of dangerously low software quality [Figure 2]. The higher the software quality the lower the risk, the sooner the products will be out to market, the fewer the defects.

4.5 Defect repair

Once the defects have been detected, their repair is facilitated by the DIS's code browser which enables fast access, navigation and querying of the code base. An added benefit of the browser is that it allows fast comprehension of complex code. These comprehension improvements deliver software quality improvements as well as increased developer productivity.

4.6 Monitor quality

Monitoring of software quality is enabled by regularly generated QFS-based quality assessments and custom trend chart generation, providing assessment of trends over time.

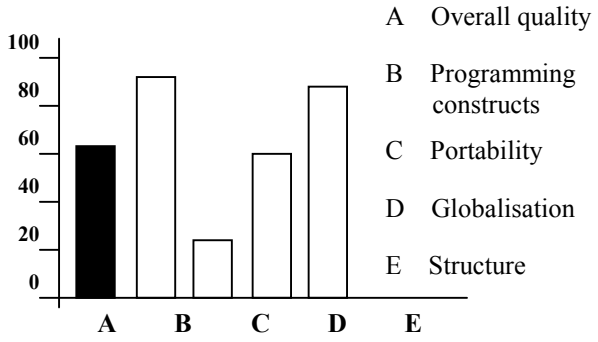


Figure 2. Quality assessment. The chart shows the overall quality index together with the quality indexes of four quality filter sets (QFS).

5 Stage 2: New Defect Prevention

5.1 Introduction

Market demands are regularly quoted as being responsible for the abandonment of sound development practices, exhorting a much higher price than most may realise. For example, an often overlooked but

extremely valuable facility to detecting defects is the use of peer reviews, or code inspections. JPL has documented a cost of only a $90 to $120 per defect if found during inspections, but an astonishing $10,000 per incident when caught in testing [1]! In the following paragraphs we go through the successive steps to enable automated defect prevention.

5.2 Impact analysis

Understanding the complex relationships between all the entities in the code base requires not only skill but also extensive experience with the specific source code. Even the best and most experienced engineers can make mistakes in their analysis of changes to the source code. For example, if an engineer changes only a single line of source in a function, there may be 15 other places in the code base that must be examined and may require related changes. Even if the developer remembers 14 of the 15 other places in the code to check, it is the one instance overlooked that can cause the code submission to fail. Such a failure impacts not only the individual developer but may very well impact the entire engineering team, or even the entire company, waiting on a successful build or defect-free release. The DIS provides reliable, automatic, accurate, precise and complete analysis of the impact of changes to the source code.

The impact of proposed entity signature changes are therefore examined *before* they become final. Impact Analysis reduces the risk of making changes before committing to them.

5.3 Submission check

Checking code for design flaws and coding defects before it is submitted to the shared source base is widely acknowledged as a good thing to do. The promise of properly conducted code reviews is that they can be effective in improving quality of software. Poor coding constructs can be identified and eliminated before they add complexity to the product. Coding defects can be found and remediated before they enter and pollute the shared source base.

However, the reality of manual code reviews usually does not match their promise. Because code reviews are labor intensive and require scarce senior developers, they are viewed by many developers as painful and a waste of time. Manual code reviews are often not done with the level of attention and energy necessary to be thorough and complete. Because of this reality, code reviews are done on an irregular basis and are frequently ineffective.

Through automation of this labor intensive and time consuming task, the quality of source code can significantly be improved prior to submission to the shared source base thereby preventing potential defects from entering the code base and affecting the entire engineering development and testing team. A *Submission Check* mechanism is implemented inside the DIS to automatically analyse and evaluate source code submissions. Submission Check examines coding constructs that cause software failures or may lead to complexity and costly rework. The submission check feature is fully customisable, through TPM and DFA technologies, to allow developers to optimise the analysis for specific coding constructs and coding standards.

6 Conclusion

The cost of software defects places a heavy burden on every software development organisation. The business need to address this problem has become critically urgent with rapidly increasing volumes of source code, rapidly increasing salaries due to fierce competition for scarce developers and rapidly changing technologies and market needs. Some companies will discover how to cope with the tremendous burden of such rapidly increasing costs…other companies will not. The two keys are; 1) Prevention and 2) Assuming all prevention is not possible, discovery of errors as early in the process as possible.

References

[1] R. Rogers, “The cost of software defects”, White paper, Software Emancipation Technology, June 1999.

[2] B. Boes, “Development Information Systems, A New Paradigm in Software Development: Complexity Begets Complexity – A Vicious Cycle”, White paper, Software Emancipation Technology, May 1999.
[3] GartnerGroup, “Software Emancipation Technology, Inc.: DISCOVER”, GartnerGroup/DataPro, Nov. 1999.
[4] B Boehm, Software Engineering Economics, Prentice-Hall, 1981.
[5] N. Gross, M. Stepanek, O. Port , and J. Carey, “Software Hell: Glitches cost billions of dollars and jeopardize human lives. How can we kill the bugs?”, Business Week Online – International, http://www.businessweek.com/1999/99_49/b3658015.htm, Dec. 1999.
[6] R. E. Park, W. B. Goethert and W. A. Florac, “Goal-Driven Software Measurement - A Guidebook”, Handbook CMU/SEI-96-HB-002003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, August 1996.
[7] P. Ntourntoufis, “From Software Quality Control to Quality Assurance”, pp. 100-4, in Embedded Systems Engineering, Vol.8 No.3, April-May 2000.
[8] P. Ntourntoufis, “Transforming Software Quality Control to Quality Assurance”, pp. 191-201, in Approaches to Quality Management, Ed. D Chadwick et al., The British Computer Society, May 2000.

QWE2000 Session 7T

Mr. Bill Lewis [Canada]
(Technology Builders, Inc.)

"Requirements-Based Testing: An Overview (Process and Techniques for Successful Development Efforts)"

Key Points

- An overview of the Requirements-Based Testing (RBT) process
- An overview of Caliber-RBT, the tool that supports the RBT process.
- The intended audience is project managers, development managers, developers, test managers and test practitioners who are interested in understanding RBT and how it can be applied to their organization.

Presentation Abstract

In many organizations, testing is viewed as slowing down the delivery of the system, partly because it is the last step in the development process. Ironically, when testing is properly deployed, with heavy emphasis on Requirements-Based Testing, it can have a major impact on overall project productivity as well as product quality.

Many organizations also have discovered that capture/playback tools require a significant investment in building and maintaining test scripts. They also discover that the scripts cannot keep up with rapidly changing specifications. This presentation will address how a Requirements-Based Testing (RBT) process provides major productivity gains, especially when used in conjunction with a tool to support it. The RBT process stabilizes the completeness of the specifications early in the development process. Caliber-RBT tool then designs an optimized set of test cases that are then automatically fed into all of the major capture/playback tools. The results are fewer tests with greater functional coverage, shortened time to delivery, reduced costs of development, and significantly improved quality.

About the Speaker

Bill Lewis has 34 years experience in the computing industry. Currently as a senior technology engineer he trains and consults in the requirements-based testing area which focuses on leading-edge testing methods and tools.

Before joining TBI, he was an assistant director for Ernst & Young, LLP for almost 6 years as the quality/ testing manager. The majority of BillÆs career was at IBM for 28 years. His jobs included system programmer, analyst, performance analyst and technical instructor. With IBM, Bill has consulted and trained all over Europe and

Asia. His first job out of college was with the Apollo Support Department for General Electric at the Kennedy Space Center as a real-time programmer.

Bill is a prolific communicator having lectured at various quality organizations including the Quality Assurance Institute (QAI) Fourth International Quality Conference, the American Society for Quality, and Association of Information Technology Practitioners. His has also authored five books on computer problem solving and has recently authored a book on software testing as a continuous quality improvement process.

Bill holds a BA degree in Mathematics from the University of Miami, Florida and an MS in Operations Research from the University of Central Florida. He is also a Certified Quality Analyst (CQA) and Certified Software Test Engineer (CTSE).

Requirements-Based Testing Overview

Bill Lewis
Senior Technology Engineer

Technology Builders, Inc.
400 Interstate Parkway
Suite 1090
Atlanta, Georgia 30339
tel: 770-937-7900
fax: 770-937-7898
http://www.tbi.com

TBI

Agenda

- The Business Case for Better Quality
- The Requirements-Based Testing Process
- Management Considerations
- Summary

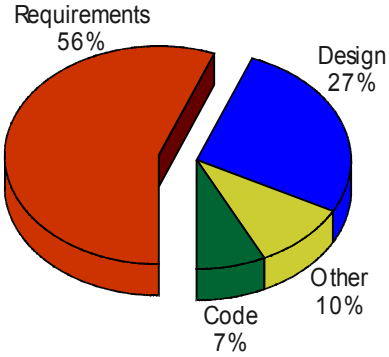
TBI

Relative Cost To Fix An Error

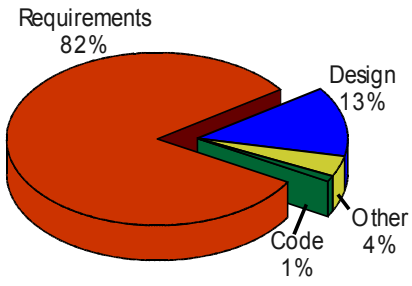
Phase In Which Found	Cost Ratio
Requirements	1
Design	3-6
Coding	10
System/Integration Testing	15-40
User Acceptance Testing	30-70
Operation	40-1000

(IBM, et. al.)

Distribution of Bugs



Distribution of Effort To Fix Bugs



(James Martin)

Why Good Requirements are Critical

Standish Group Statistics for 1997

- In 1997, American companies spent $100 BILLION for canceled software projects.
- $45 BILLION spent for projects which significantly exceeded time and budget estimates.

Standish Group Statistics for 2000

- In 2000, American companies will spend $84 BILLION on failed software projects
- $192 BILLION will be spent on projects that significantly exceed time and budget estimates, or have reduced functionality

5

Why Good Requirements Are Critical

Top reasons for failure:

- Incomplete requirements and specifications
- Changing requirements and specifications
- Lack of user input

(Standish Group and other studies)

6

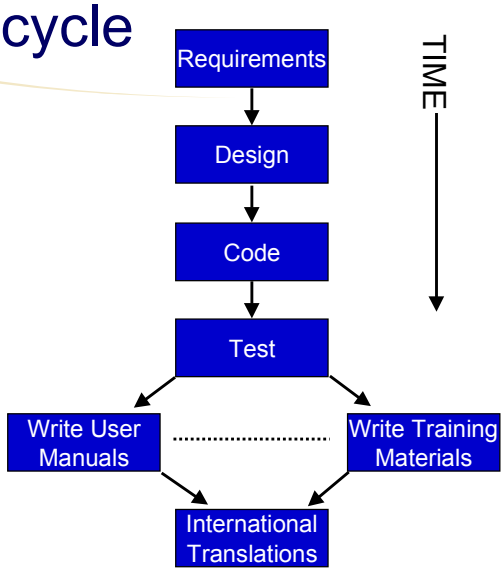
The Test Process - Make It:

- Timely:
  - Integrate throughout the life cycle
- Effective:
  - Rigor in test definition
- Efficient:
  - Heavily automated
  - Minimum number of tests
- Manageable:
  - Measurable
  - Predictable

7

TBI

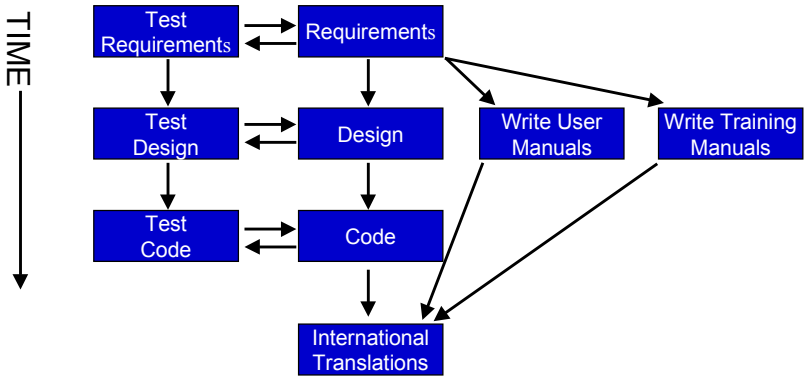
Standard Development Lifecycle



8

TBI

Lifecycle With Testable Requirements and Integrated Testing



9

TBI

The Requirements-Based Testing Methodology

Quality filters

1. Validate requirements (WHAT) against objectives (WHY)
2. Apply use-cases against requirements
3. Perform initial Ambiguity Reviews
4. Perform domain expert reviews
5. Create Cause-Effect Graphs
6. Logical consistency check and test cases designed by Caliber-RBT
7. Review of test cases by requirement authors
8. Review of test cases by users/domain experts
9. Review of test cases by developers
10. Walk test cases through design
11. Walk test cases through code
12. Execute test cases against code

10

TBI

Characteristics of a Testable Requirement

1. Deterministic
2. Unambiguous
3. Correct
4. Complete
5. Non-redundant
6. Lends itself to change control
7. Traceable
8. Readable by all project members
9. Written in a consistent style
10. Processing rules reflect consistent standards
11. Explicit
12. Logically consistent
13. Lends itself to re-usability
14. Terse
15. Annotated for criticality
16. Feasible

Ambiguity Review Checklist

- Dangling else
- Ambiguity of reference
- Scope of action
- Omissions
  - Causes without effects
  - Missing effects
  - Effects without causes
  - Complete omissions
  - Missing causes
- Ambiguous logical operators
  - Or, And, Nor, Nand
  - Implicit connectors
  - Compound operators
- Negation
  - Scope of negation
  - Unnecessary negation
  - Double negation
- Ambiguous statements
  - Verbs, adverbs, adjectives
  - Variables, unnecessary aliases
- Random organization
  - Mixed causes and effects
  - Random case sequence
- Built-in assumptions
  - Functional/environmental knowledge
- Ambiguous precedence relationships
- Implicit cases
- Etc.
- I.E. versus E.G.
- Temporal ambiguity
- Boundary ambiguity

Test Case Design Approaches

Goal:

Design a necessary and sufficient set of test cases to ensure system integrity.

- Production files
- Gut feel
- Exhaustive “Combinatorics” of inputs
- Rigorous algorithms

13

TBI

Testing With Only Production Files

- Covers less than 30% of the code
- Exception cases not covered since data is already scrubbed
- Time-dependent functions not covered
- Expected results not determined for every output field
- Might find some missing cases
- Have value in performance testing
- Have value in helping build test cases

14

TBI

Testing By Gut Feel

Totally dependent on who is doing the testing:

- How experienced they are at testing
- How experienced they are in the application
- How experienced they are in the technology that the application runs on
- How they are feeling today

Even if all the tests run successfully, all you know is that *those* tests run -- not that the system runs successfully

Testing By Brute Force Combinatorics

Amount of time to arrange 15 books in every possible way:

2,487,996 YEARS at one change per minute

Length of paper required to write all the possible sequences of the 26 letters of the alphabet:

160 million light years

QUESTION: how many variables are in <u>your</u> application?

CALIBER RBT

- Validates requirements
- Provides maximum coverage with the minimum number of tests
- Automates test case design
- Uses a rigorous algorithm

17

TBI

Designing Test Cases -- An Example

- This function has sixty-four possible combinations of input from which to select test cases:

  If the customer is a business client or a preferred personal client and they have a checking account, $100,000 or more in deposits, no overdraft protection and fewer than 5 overdrafts in the last 12 months, then set up free overdraft protection. Otherwise, no overdraft protection.

- How many test cases are required to confirm that the function works?

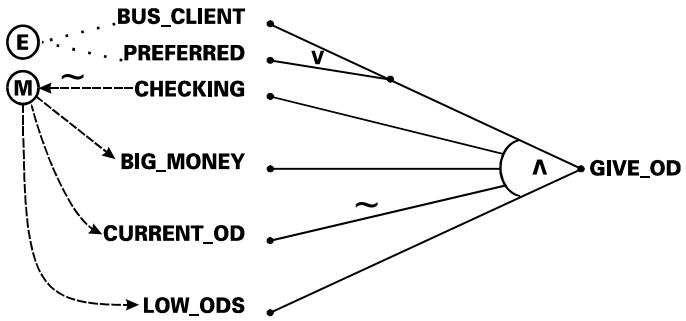
  Answer: Seven

  Question: Can you name them?

18

TBI

Cause-Effect Graph



Test Cases Designed By Caliber-RBT

```
TEST#1 -- Automatic Check For Overdraft Protection

Cause states:
    The customer is a business client
    The customer has a checking account
    The customer has $100,000 or more in deposits
    The customer does not have overdraft protection
    Overdrawn less than five times in last 12 months


Effect states:
    Set up free overdraft protection
```

Caliber-RBT takes the cause-effect information entered and designs a complete set of TEST CASES.

```
Automatic Check For Overdraft Protection
STATISTICS

Synthesis of NEW Tests was specified.

There were 14 input statements processed.

There were 9 Functional Variations generated.
    There were NO Infeasible Variations
    There were NO Untestable Variations

There were 7 NEW Test Case definitions generated by Caliber-RBT.
    There were 9 Tested Variations, and
    9 FEASIBLE Variations, which yields
    100% Functional COVERAGE of FEASIBLE variations.

There were 9 Tested variations, and
    9 TESTABLE variations, which yields
    100% Functional Coverage of TESTABLE variations.

For n = 6 Primary causes, then
    2^n = 64 THEORETICAL Maximum Number of Test Cases.

Caliber-RBT generated 7 Test Cases, which yields a
    9 to 1 Test Case Compression Ratio.

Caliber-RBT processed 9 Testable Variations, which yields a
    1 to 1 Testable Variations to Test Case Compression Ratio.


Caliber-RBT Elapsed Time:  00:00:01 (hh:mm:ss)
```

Summary statistics are produced to aid in project estimating and tracking.

21

Test Statistics For a Large Problem

```
CHP_PG5_26/TOBACCO USE STATISTICS


Synthesis of NEW Tests was specified.

There were 112 input statements processed.

There were 141 Functional Variations generated.
    there were NO infeasible Variations
    there  was 1 Untestable Variation

There were 22 NEW Test Case definitions generated by Caliber-RBT.
    there were 140 Tested Variations, and
    141 FEASIBLE Variations, which yields
    99% Functional COVERAGE of FEASIBLE variations.

There were 140 Tested variations, and
    140 TESTABLE variations, which yields
    100% Functional Coverage of TESTABLE variations.

For n = 37 Primary causes, then
    2^n = [a little more than] 137,438,953,472 THEORETICAL Maximum Number of Test Cases.

Caliber-RBT generated 22 Test Cases, which yields a
    6,247,225,157 to 1 Test Case Compression Ratio.

Caliber-RBT processed 140 Testable Variations, which yields a
    6 to 1 Testable Variations to Test Case Compression Ratio.


Caliber-RBT Elapsed Time:  00:00:02 (hh:mm:ss)
```

22

Justification for Rigorous Testing

- Thought experiment 
  - Put 137,438,953,450 **RED** balls in this room
  - Add 22 **GREEN** balls to the room and mix well
  - Turn out the lights
- Pull out 22 balls 
  - What is the probability that you have selected the 22 green ones?
- Pull out 1,000 balls
  - What is the probability that you have the 22 green ones now?
- Pull out 1,000,000 balls
  - What is the probability that you have the 22 green ones now?

** This is what "GUT FEEL" testing really is.**

TBI

Would You Rather Validate a Specification That Looks Like This:

James Joycian Novel Style

Dental Insurance Claims Payment Specification

Dentists with membership codes of 2, 3, or 9 are member dentists. For claims referencing a non-member dentist or for procedures not within the referenced dentist's record, a system table is used to calculate the amount paid. Otherwise, the amount submitted is paid. However, an override code of 1 or 9 allows the amount submitted to be paid for non-member dentists or for procedures not within the referenced dentist's record. When an override code is used an entry is made on the paid claims report.

TBI

Or Validate Test Cases That Look Like This:

TEST 1	TEST 2
Cause States: The Dentist is a Member Dentist The procedure was not preauthorized An override code was entered Effect States: This is a potential partial payment situation Override the partial payment Pay the full amount of the claim Make an entry on the paid claims report	Cause States: The Dentist is a Member Dentist The procedure was preauthorized Effect States: It is a valid procedure for the member dentist Pay the full amount of the claim Do not make an entry on the paid claims report

25

Management Considerations

- Staffing curve peaks earlier
  - Requirements are written in more detail
  - Design is concurrent with requirements
  - Implementation preparation is concurrent with design
  - Testers are involved from the beginning
  - Technical writers are involved earlier
- Total resources reduced
  - Minimize scrap and rework
  - Plans have better focus on scope and priorities

26

RBT Process
Test Case Design/Execution Status Report

Function Priority	Function Name	Requirements Written	Ambiguity Review Complete	Requirements Complete	Cause-Effect Graphs	Caliber-RBT Entry	Test Cases Designed	Test Cases Reviewed	# of Functional Variations	# of Test Cases	Modules Coded	Test Cases Built	Test Cases Executed	# of Defects Identified	# of Defects Open	% Code Coverage
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
	TOTALS															

72

TBI

Summary
What Caliber-RBT Delivers:

- *Maximum* coverage with *minimum* tests
  - 100% functional coverage
  - 70-90% code coverage
- Quantitative test progress metrics
- Testing is no longer a bottleneck
- Highly portable test scripts
- Tests any application written in any language running on any computer

28

TBI

QWE2000 Session 7A

Ms. Jill Pritchet & Mr. Ian Lawthers [Ireland]
(Centre for Software Engineering)

"Software Process Improvement for Small Organizations using the "SPIRE" Approach"

Key Points

- Outlines a comparatively low cost approach to Process Improvement particularly suitable for small to medium sized software companies
- Presents results and analysis of companies that have used the approach
- Describes the future direction of the work

Presentation Abstract

Shows how a focussed, pragmatic approach to process assessment and improvement has yielded real benefits to a number of small to medium sized companies involved in developing software in four European countries.

The paper describes the Software Process Improvement in Regions of Europe (SPIRE) project. SPIRE was a project funded through the European Systems and Software Initiative (ESSI), which started in March 1997 and ran for two years. SPIRE's objectives were to lower the barriers preventing Small Software Developers (SSD s - companies/departments of companies with fifty or fewer software staff) from successful participation in SPI by:

- raising awareness of SPI benefits among decision makers and change agents in SSDs

- educating participating SSD managers and staff in practical SPI skills

- helping SSDs to maintain momentum in carrying through their improvement plans

The SPIRE project used the SPIRE Approach, a low cost way to help small software companies to implement their first SPI projects efficiently. The paper will explain what the SPIRE Approach is, and how it was used in the project.

About the Speaker

Jill Pritchet is a Senior Consultant at the Centre for Software Engineering in Dublin, Ireland. She is responsible for promoting Software Process Improvement and Quality in Irish Software Companies through awareness and training programmes,

consultancy support and relevant European programmes. She is also the Project Manager for the Software Process Improvement in Regions of Europe (SPIRE) project.

QWE 2000

SPI for Small Organisations using the "SPIRE Approach"

Presented by: Jill Pritchet,
Centre for Software Engineering, Ireland

Presentation Objectives

- Introduce the SPIRE project
- Explain the role of the SPIRE Mentors
- Introduce the outputs from the project
  - SPIRE Handbook & Case Studies
- Explain the benefits of process assessment
- Explain the "*SPIRE approach*" to SPI
- Outline some key results from the project
- Outline the future plans for SPIRE

Background to the project

- CEC funded ESPRIT/ESSI project, with partners in 5 countries:
  - *Austria - ARC Seibersdorf; Ireland - CSE, Dublin; Italy - Etnoteam, Milan; Sweden - IVF, Gothenburg; UK - SIF, Belfast*
- Objective: to lower the barriers preventing SSDs from successful participation in SPI projects

SSD: a small software development unit with up to 50 software people, both small software companies and small software units in larger user companies

How was this done?

- By convincing decision makers and change agents of the benefits of SPI
- Educating participating SSD managers & staff in practical SPI skills
- By using the "*SPIRE Approach*"
- Providing guidance from experts experienced in SPI, in the form of Mentors paid by SPIRE
- Helping SSDs to maintain momentum in carrying out their improvement project

The role of the Mentor

- Helped the SSDs to:
  - carry out an assessment of their needs
  - prepare a sound project plan for a cost-effective, small SPI project (funded up to 15K Euro)
  - implement the project
  - evaluate the project results
  - have confidence to apply the techniques on their own later

The SPIRE Handbook

- The SPIRE Partners published the SPIRE Handbook (now re-published) as a guidebook
- Purpose of the book:
  - to provide a practical guide for those considering SPI
- The book is split into 3 Parts:
  - Part 1 - Business Managers guide to SPI
  - Part 2 - Champions guide to SPI
  - Part 3 - Software Process and Best Practice Framework

Part 1 - Business Managers guide to SPI

- For the decision makers whose leadership and informed commitment of resources (*money & labour*) is critical for SPI success
- It explains:
  - the basics of SPI
  - how to make a business case for investment in SPI
- It also gives guidance on how to support and foster successful SPI

Part 2 - The Champions guide to SPI

- For the management change agent, the person with responsibility for leading and managing the improvement project
- It explains:
  - how to follow the steps of a cycle of continuous improvement
  - how to handle the people and cultural issues which impact success
  - how to manage the improvement project

Part 3 - Software Process and Best Practice Framework

- For the Managers and Software Engineers to analyse the component processes and to identify industry best practices to strengthen them
- It explains:
  - the purpose of each process
  - what it should cover
- It also gives guidance in the form of helpful hints and pitfalls to avoid

Process Assessment

- SPIRE used scaled down SPICE assessments
- The assessments were carried out twice:
  - at the start, to assist the SSDs to focus their improvement project in the most beneficial area, based on the business needs of the organisation
  - at the end of the projects, to confirm the level of improvement achieved

Process Assessment *continued*

- Automated tools were used to make the assessments quicker and easier
  - Bootcheck
  - Synquest
- The assessments were a low cost, but effective mechanism for measuring the success of individual projects

The SPIRE Approach

- The approach is an 8-step process used by all the SSDs
- SSDs found it was an easy to understand approach
- It helped them to get the most out of their projects

Full detail can be found in the SPIRE Handbook but to summarise the steps are:

The steps of the SPIRE Approach

Step 1 - Examine organisation needs
- so that you focus on the most beneficial area

Step 2 - Initiate process improvement
- treat it as a project and plan the improvement

Step 3 - Prepare and conduct process assessment
- consider appropriate tools

Step 4 - Analyse results and derive action plan
- start with something simple and quantify your goals

The steps of the SPIRE Approach

Step 5 - Implement improvements
- consider a pilot rather than “big bang” approach

Step 6 - Confirm improvements
- Was it a success? Do a second assessment.

Step 7 - Sustain improvement gains
- Systematically collect data to track project performance

Step 8 - Monitor performance
- Fine tune your skills

SPIRE CASE STUDIES

- To help others the most successful SPIRE projects prepared Case Studies which are freely available from the Web site: *http://www.cse.dcu.ie/spire*
- They are in a variety of languages & short and easy to read
- They can be used as a valuable source of information on real experiences
- Can help to convince Managers of the benefits of SPI in small companies

Example Case Study - Cunav Technologies
(now NewWorld Commerce)

- Software Systems development and consulting company
- Focus was in improving their requirements analysis process
- Objectives were to:
  - improve the ability to manage customer expectations
  - deliver systems with significantly reduced need for rework

Example Case Study continued

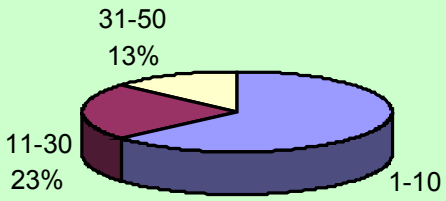
- What was achieved:
  - a requirements analysis process was developed
  - training in requirements gathering was provided for their consultants
- Several key areas were improved:
  - a decrease of 90% in the amount of rework required
  - number of requirements related bugs fell
  - accuracy of time and budget estimates improved

Statistics

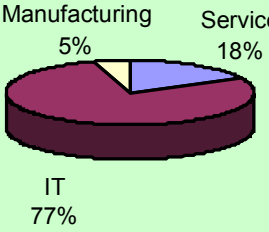
To give you a feel for the project here are some relevant statistics taken from the European Analysis Report produced at the end of the SPIRE project.

Note: The full report is available from the SPIRE web site

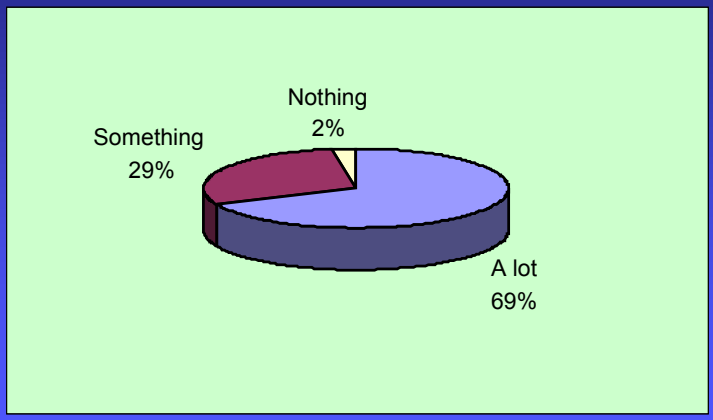
Size of Software Production Unit



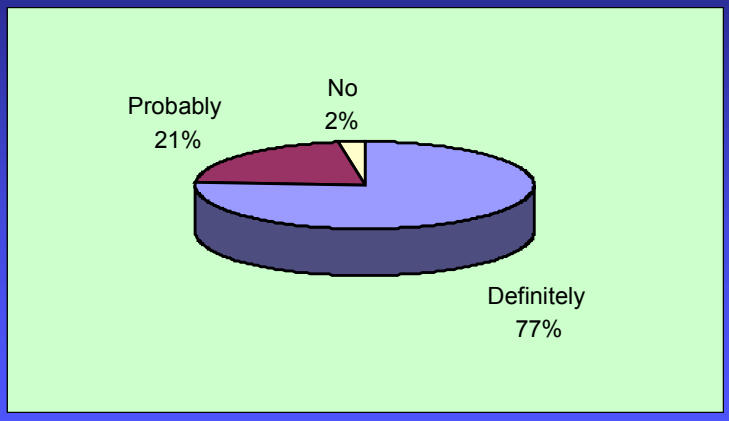
SSDs by industry sector



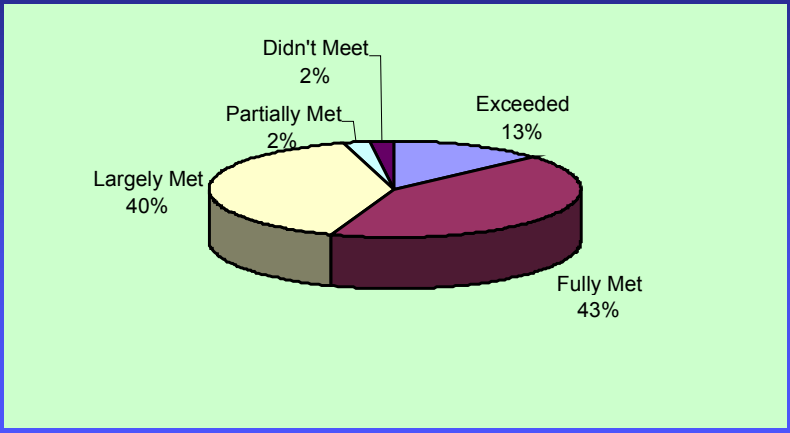
Extent to which SPI Skills can be Applied in the Future



Would the SSDs do another Improvement Project Without External Funding?



Were the Objectives of Improvement Plan Met by the SSDs?



Staff Attitude Change

For all Regions:

The staff attitude score at the start averaged 0.58

The staff attitude score at the end averaged 1.02

There was a positive increase of +0.44

Therefore the SPIRE projects did not have a negative impact on the participating companies staff

Achievements

- Major impact on the 73 SSDs that took part
- They achieved worthwhile improvements in their processes
- Improved their business and competitive position
- Management & key staff educated in SPI skills which are being used to make further improvements after the project ended

Future Plans

- Two of the original partners are continuing the good work done with SPIRE:
  - CSE, Dublin, Ireland and Arcs Seibersdorf, Austria
- They are:
  - setting up Training and Consulting for Software Companies using the SPIRE Approach
  - doing this in their own regions and also other parts of Europe and North America
  - looking at sub licensing the materials through suitable agencies

CSE plans

Preparing a 1-day SPI for Managers course

- aimed at convincing Managers of Small Software Organisations that initiating an SPI project is not only viable but cost justifiable
- based on Part 1 of the SPIRE Handbook

CSE plans continued

Developing a SPIRE Coaching Cluster

- group of companies attend training days to learn the SPIRE approach
- mentor assigned to help them do an assessment followed by implementation of a focussed SPI project in-house
- based on Part 2 of the SPIRE Handbook

Both courses planned to be available in first quarter of 2001

Arcs Seibersdorf Plans

- Joint training venture, mainly in the German speaking regions, using the SPIRE technology *(including tools & technologies of the Partners e.g. ESI, Bootstrap Institute)*
- Investing in a Benchmark service, balanced score cards and a qualification for SPIRE mentors
- SPIRE being included as part of the post-graduate MAS studies at the Danube University

Conclusions

- The SPIRE project has had a major impact in:
  - raising the awareness of the benefits of SPI to small software organisations
  - providing real evidence that SPI is viable in these operations
  - educating a significant number of SSDs to a level where they are able to continue SPI on their own
  - raising the capability levels of the participant companies

Conclusions

Proving that SPI can be effective in many different areas such as:

- Software Testing
- Configuration management
- Subcontractor management
- Project management
- GUI development
- Web site development
- Standard procedures
- Introduction of PSP
- Software Life Cycle process model
- Requirements analysis etc.,

Recommendations

- Considering an SPI initiative? - look at the compelling evidence from SPIRE
- Data is now available for successful SPI in small organisations
- If funds allow consider using a Mentor
- Get a copy of the SPIRE Handbook (only 25 Euro) it will help you to carry out your first SPI project *(see me or order from the web site http:/www.cse.dcu.ie/spire)*

Finally

Go on give it a try, you have nothing to lose, and you may be surprised at how much you can gain in a very short time frame!

Contact Information

Centre for Software Engineering
Jill Pritchet or Ian Lawthers
DCU Campus,
Glasnevin,
Dublin 9, Ireland

Tel: +353 1 700 5750
Fax: +353 1 700 5605
Email: jill@cse.dcu.ie or ian@cse.dcu.ie

ARCS (Austrian Research Centers Seibersdorf)
Erwin Schoitsch
Information Technologies
Forschungszentrum
A-2444 Seibersdorf, Austria

Tel: +43 2254 780 3117
Fax: +43 2254 72133
Email: erwin.schoitsch@arcs.ac.at

Software Process Improvement for Small Organisations using the "SPIRE Approach"

Author: Jill Pritchet and Ian Lawthers
Centre for Software Engineering, Dublin City University Campus, Dublin 9, Ireland

1. Summary

The paper shows how a focussed, pragmatic approach to process assessment and improvement has yielded real benefits to a number of small to medium sized companies involved in developing software in four European countries.

It describes the ***Software Process Improvement in Regions of Europe (SPIRE) project***. SPIRE was a project funded through the European Systems and Software Initiative (ESSI), which started in March 1997 and ran for two years. SPIRE's objectives were to lower the barriers preventing Small Software Developers (SSDs - companies/departments of companies with fifty or fewer software staff) from successful participation in SPI.

The SPIRE project used the ***SPIRE Approach***, a low cost way to help small software companies to implement their first SPI projects efficiently. The paper will explain what the ***SPIRE Approach*** is, and how it was used in the project.

2. Introduction

When the 5 Partners set up the SPIRE project they recognised that in order for the project to be a success they needed to find ways to enable SSDs to lower the barriers that were preventing them from successful participation in SPI. This was achieved in a number of ways by:

- Convincing decision makers and change agents of the benefits of SPI
- Educating participating SSD managers and staff in practical SPI skills, that they would retain after the project ended
- Helping SSDs to maintain momentum in carrying through their improvement plans
- Providing guidance from experts experienced in SPI in the form of Mentors paid for by the SPIRE project funding

The SPIRE project adopted the ***SPIRE Approach***, which proved to be a very successful way of helping small software companies to implement their first SPI projects.

3. Results and achievements

SPIRE had a major impact on the 73 or so SSDs, which undertook focused process improvement projects under the guidance of the SPIRE mentors. The impact was in two ways:

- Firstly they achieved worthwhile improvements in their processes, which not only improved their business but also their competitive position

- Secondly their management and key staff have been educated in the skills of practical process improvement, with most of them applying the skills to make further improvements after SPIRE finished.

4. The use of Mentors in SPIRE

Since the majority of participating SSDs had never tried an SPI project before; SPIRE paid for experienced mentors to help guide them through each of the following:

- An assessment of their needs, to ensure they focused on an area for improvement that would add value to their business, in a short time frame.
- The preparation of a sound project plan for a cost-effective, small SPI project (funded by SPIRE to a maximum of 15K Euro).
- Implementation of the project, to ensure success.
- Evaluation of results, to give them the confidence to approach a similar activity on their own, once SPIRE was over.

The use of Mentors was seen as one of the most significant factors in ensuring the success of the projects.

5. Handbook

During the SPIRE project the Partners published the very successful ***SPIRE Handbook***, which is a guidebook, with the clear objective of meeting the practical needs of busy managers and software engineers, working in real-world businesses.

At the time demand for the book far exceeded supply and as a result two of the SPIRE Partners have recently re-published the Handbook.

The Handbook is split into 3 sections:

Part 1 is a Business Manager's Guide to SPI. It addresses the management decision maker, who may not even be a part of the software organisation in a software user company, but whose leadership, and informed commitment of resources, both money and labour, is critical for SPI success. It explains the basics of SPI, and how to make a business case for investment in SPI. And it gives guidance on how to support and foster successful SPI.

Part 2 is a Champion's Guide to SPI. It addresses the management change agent, who will take, or be given, responsibility for leading and managing the improvement project. The champion has a key role in making improvement happen. Without a highly motivated individual, acting with the support of business management, to convey the vision of improvement and find ways round the barriers to change, the chances of success are much reduced. This part explains how to follow the steps of a cycle of continuous improvement, how to deal with the people and cultural issues which impact success, and how to manage an improvement project.

Part 3 is a Software Process and Best Practice Framework. It helps the Managers and Software Engineers making up the improvement project teams, to analyse the component processes of their overall software process, and to identify industry best practices to strengthen them. It explains the purpose of each process, what it should cover, and gives guidance in the form of helpful hints and pitfalls to avoid. It is a comprehensive framework, in that it examines in turn all the processes that you might wish to improve.

6. Process Assessment

Another key factor that influenced the success of the SPIRE projects was the use of scaled down ***SPICE process assessments*** (SPICE – Software Process Improvement Capability dEtermination Standard - ISO/IEC TR 15504) for small organisations.

The assessments were carried out twice, once at the start of the project, in order to assist the SSDs to focus their SPI improvement project in the most beneficial area based on the business needs of the Organisation. The second assessment was carried out at the end of the project to confirm the level of improvement achieved. This proved to be a very low cost (but effective) mechanism for measuring the success of individual projects.

To make the process of assessment as easy as possible the SPIRE Partners used an automated tool, selected from either ***Bootcheck*** or ***Synquest***.

7. The SPIRE Approach

The ***SPIRE approach*** is an 8-step process, which was used by all the SSDs that participated. They found that it was easy to understand and that it helped them to get the most out of their first SPI projects. The full detail can be found in the SPIRE Handbook but in summary the 8 key steps are as follows:

Step 1 – Examine organisations needs

When you start a process improvement initiative for the first time it is an ideal opportunity to steer managers and employees to discuss and then decide on which business objectives the organisation intends to achieve. Failing to define such objectives could easily lead to a failure of the entire improvement effort and to a waste of the company's efforts. An easy approach to begin defining the organisation's objectives is the ***SWOT analysis***, which is an evaluation of the organisation's strengths, weaknesses, opportunities and threats.

Step 2 – Initiate process improvement

The only sure way to succeed with an SPI initiative is to treat it as a project in its own right. It is essential that you establish proper Project Planning, which is then managed in an efficient and effective manner during the life of the project. The organisation must clearly define the objectives, scope, budget and time constraints for the SPI initiative.

Step 3 – Prepare and conduct process assessment

If you don't know where you started from how will you know that you have moved forward? This is one of the key questions we asked each SSD to consider and to enable them to answer *"where they were starting from"* we asked their Mentors to help them undertake a process assessment.

Several assessment methods are available; the SPIRE project used the SPICE model, which provides an accurate assessment of software-specific processes, and other processes that support software development and maintenance.

A report is produced from the assessment, which can then be compared with a later assessment carried out at the end of the project to establish the level of improvement achieved.

In parallel with the SPICE assessment the SSDs, and their mentor, also completed a staff attitude survey. The objective of the survey was to ascertain the level of commitment from staff, which can have a major affect on the project success.

Step 4 – Analyse results and derive action plan

This step involves identifying which software processes are critical for the achievement of business goals. This is done by ranking the processes according to their relevance to business needs and to their capability level: processes that are very relevant to business needs and get a low capability reading are those with highest priority to improve, and where investments will be most beneficial.

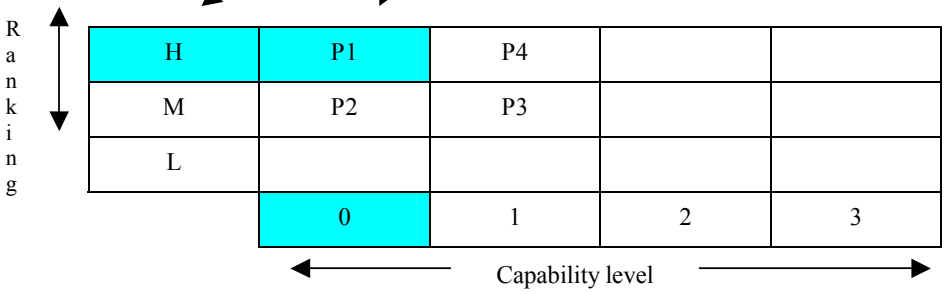
In SPIRE the SSDs, with the help of their mentor, ranked their processes as either L (low) M (medium) or H (high). They then matched this against their assessment results to highlight the areas to consider for improvement.

In the example below Process P1 has been assessed at capability level 0 bit it is highly relevant for the SSD in trying to achieve Business Need 1. This makes it an ideal choice as a potential focus for an improvement project.

We recommend that you start with simple improvements that you are confident will achieve results, rather than trying complex ones at your first attempt.

Example : The table below shows you how the results might be represented:

Business Needs	Processes supporting achievement of business needs	Relevance (H,M,L)
Need 1	Process 1	H
	Process 2	M
	Process 4	L
Need 2	Process 3	M
	Process 4	H
Need 3	Process 1	M
	Process 5	L
etc	etc	etc



When you specify the improvement goals for the selected processes try to do so quantitatively e.g. 20% reduction in failure rate or 10% improvement in timeliness. This will make them easier to understand. It is also a good idea to validate the results of the improvement actions against the initial goals using appropriate metrics.

Other things to consider in this step are the appropriate allocation of resources and the identification of actions to be carried out to improve the target processes e.g. definition of template or definition of steps for requirement specification etc.,

Step 5 – Implement improvements

The implementation can be done in a number of different ways depending on the scope of the improvement project. In some instances a pilot might be a beneficial way of experimenting with SPI. When the pilot has been a success it is much easier to roll it out to the rest of the organisation.

You can of course define the process changes and then apply them to the whole organisation in one go, but be careful, as this could cause more disruption to the organisation, which may already be under pressure.

At this point it is appropriate to consider the risks associated with your selected project. Document the risks in the Project Plan and what you propose to do to minimise these risks. Ensure you have Senior Management commitment to the improvement project before continuing.

If you have used a Mentor they will be able to assist you with the implementation of your project. You (*or you and your Mentor*) must monitor and track progress of the improvement activities. As with any project as soon as you realise a problem has occurred that affects progress you must take appropriate action. Re-planning may be required.

Capture the measurement data identified in the previous stage to be used for evaluation of results and achievement of targets.

Step 6 – Confirm improvements

Now is the time to look back at what happened and to ascertain if your improvement project was a success. Were the targets you set reached? Did the benefits you expected come to fruition? Is the cost/ benefit ratio satisfactory?

We recommend that you also do a second assessment to evaluate whether the planned capability improvements have been achieved. As before repeating the staff attitude survey may prove beneficial but remember to compare like with like, if the staff that originally were assessed are no longer in the organisation you can not compare earlier results against the new ones!

Discuss the results with your Senior Management.

Step 7 – Sustain improvement gains

Now is the time to ensure that the benefits achieved can be maintained now that the project has finished. Make sure that everyone, to whom it is relevant, is using the improved process.

If you took the pilot project approach you can now extend it to other parts of the organisation. Remember to plan the roll out, including assigning appropriate resources, to ensure success. You may need to consider training needs, when to

implement, how you are going to validate the implementation and how you will monitor results.

To ensure that the improved process will be followed and expected benefits realised, systematically collect and use data to track project performance. Keep measurement data simple and aligned to organisation business goals.

Step 8 – Monitor performance

Process improvement is a continuous activity, which supports your organisation to evolve to meet its business goals. Therefore at the end of one improvement project is the ideal time to think about the organisations objectives and to plan what to improve next. Sometimes the first experience of process improvement can be time consuming it is only with practice that you can get better at it. By reviewing each improvement project on completion to identify what went well, and what could have been handled better, that you will fine tune your skills, this is also a part of process improvement!

8. Case Studies

The experience gained in the most successful SPIRE projects were published as short Case Studies aimed at decision makers in SSDs in 6 languages (German, Italian, English, French, Spanish and Swedish). Data from all the projects was gathered in a standardised way, to permit analysis from which valuable lessons regarding best SPI practice for SSDs was derived and published as a report.

The results have been disseminated on paper, electronically and through workshops, in the four participating regions, which were Ireland (North and South), Italy, Austria and Sweden, and throughout Europe.

Those involved in the SPIRE projects felt that it was very worthwhile to them personally as well as their organisation. The majority felt that they would now be confident enough to undertake another SPI project internally without the funding from the Commission. They felt there was sufficient ***real evidence*** now to convince Senior Management of the benefits of SPI.

To illustrate this I have included below an overview from two of the SPIRE Case Studies. The full set of SPIRE Case Studies can be downloaded from the SPIRE web site: http://www.cse.dcu.ie

Case Study 1 – Cunav Technologies *(now NewWorld Commerce)*

Cunav Technologies is a software systems development and consulting company, which provides IT resources and solutions to customers operating in a variety of application areas. As we are continually involved in the development of specialised software systems on behalf of our clients, an ability to elicit precise system requirements from our customers obviously has a significant impact on our business.

Specifically, we saw that **improving our requirement analysis process** would improve our ability to manage customer expectations and to deliver systems with significantly reduced need for rework. As a result of our SPIRE project, a requirement analysis process was developed and training in requirements elicitation provided for Cunav consultants.

The initiative was very successful, as evidenced by improvements in several key areas of our development process. The amount of rework and number of requirements

related bugs have both fallen and the accuracy of our time and budget estimates has improved. Most importantly, we are in a position to understand the needs of our customers and deliver top quality systems on time and within budget.

The most important thing learned is that software processes are fundamental to the smooth running and success of a growing company such as Cunav. Software requirements in particular are critical elements of any system and having an efficient, consistent and cost effective means of handling these is of paramount importance.

Case Study 2 – Peregrine Systems Ltd.,

Peregrine Systems Ltd. is a successful and fast growing software development company which provides a specialised range of Business Process Automation application software products to clients in the financial services and government industry sectors.

The company's main business objective is the development and growth of its customer base. The impact of this growth is that the organisation is becoming increasingly involved in larger projects, therefore increasing the size of software development teams. Management at Peregrine became aware that their current approach to source code management would not be adequate to handle this change. It was becoming increasingly necessary to introduce source code management procedures and also the tools to support them. Additionally, a requirement was identified for an improved defect reporting and tracking system.

The purpose of this project was to improve existing procedures for source code management and defect control, and to identify and implement suitable tools to support these procedures. The aim was to provide an improved level of service to an increasing customer base.

The goals of the project were achieved over the period of the improvement project. Tools to support source code management and defect control were evaluated, selected, purchased and installed. In addition, appropriate procedures were developed and implemented.

Peregrine's capability in the areas of source code management and problem resolution has increased significantly over the life of the project. This enabled the company to handle the added burdens of growth in their customer base.

9. Future Plans

Since the end of the SPIRE Project two of the original Partners have been continuing the good work done in SPIRE. The two Partners are:

Centre for Software Engineering Jill Pritchet or Ian Lawthers DCU Campus, Glasnevin, Dublin 9, Ireland Tel: +353 1 700 5750 Fax: +353 1 700 5605 Email: jill@cse.dcu.ie or ian@cse.dcu.ie	ARCS (Austrian Research Centers Seibersdorf) Erwin Schoitsch Information Technologies Forschungszentrum A-2444 Seibersdorf Austria Tel: +43 2254 780 3117 Fax: +43 2254 72133 Email: erwin.schoitsch@arcs.ac.at

These Partners have been setting up Training and Consultancy for Software Companies in SPI using the ***SPIRE Approach***. Each Partner is handling training in their respective Regions as well as allocated Regions of Europe and North America. The Partners will be looking, in the longer term, at sub licensing the SPIRE materials in other Countries, if suitable interested agencies can be found.

Below is some detail about what each Partner is currently working on, for more information contact them directly.

Centre for Software Engineering (CSE)

CSE is developing training in two different formats. The first is a one-day course titled ***"SPI for Managers"*** this course is aimed specifically at convincing the Managers of Small Software Companies that initiating a Software Process Improvement project is not only a viable option but is also a cost justifiable one. The course stresses the need for business leadership to ensure success and is based around Part 1 of the SPIRE Handbook.

The second will take the form of a ***"SPIRE Coaching Cluster".*** CSE has used clusters for many years and has found it to be an ideal format for helping small companies to achieve real results. A group of companies will attend training days away from the office to learn an aspect of the SPIRE Approach. They will then be given assistance from a Mentor to help them with an assessment followed by implementation of a focussed SPI project in-house. Emphasis is given to the people and cultural aspects of achieving a successful improvement project. This is based on Part 2 of the SPIRE Handbook.

The courses are planned to be available in the first quarter of 2001.

Arcs (Austrian Research Centers Seibersdorf)

ARCS is creating a group of partners (joint venture) mainly in the German speaking regions to exploit the SPIRE technology (in combination with tools and technologies of the partners, ESI (European Software Institute) and BI (Bootstrap Institute)) both directly and as a franchising business in Central Europe.

As well training and tools to support the SPIRE Assessments and Improvement Processes Arcs is investing in a benchmark service, balanced score cards and a qualification, licensing and certification scheme for "SPIRE mentors". Later on ARCS staff members active in the area of academic research will present the SPIRE method as part of the post-graduate MAS-studies at the Danube University in Krems/Lower Austria.

10. Conclusions

The experience and results generated and disseminated by SPIRE have had a major impact in raising the awareness of the benefits of SPI in a significant proportion of the 100,000 or so European SSDs. The impact has been seen particularly within the 4 SPIRE regions (Austria, Ireland, Sweden and Italy), where it substantially increased the proportion of SSDs undertaking SPI projects, but has also extended right across Europe.

Specifically SSDs benefited by:

- Greater awareness of SPI and how beneficial it can be when done the right way
- Achieving worthwhile improvements in their processes, which has improved their business and competitive position
- Educating their management and staff in the skills of practical process improvement, with most of them planning to apply the skills to make further improvements after the SPIRE project finished.
- Significant increases in the capability level of the processes they targeted on the SPICE scale
- The effectiveness of the mentoring which helped stimulate them into taking SPI action.
- Proving that SPI projects can be successful in many different areas, such as:

Software Testing
Configuration Management
Subcontractor Management
Project Management
GUI Development environment
Standard Procedures
Web Site development
Introduction of PSP
Software Life Cycle Process Model
Requirements Analysis
Quality System Management

In addition the Mentors involved in the SPIRE projects benefited by:

- Direct experience gained from supporting the SSDs in implementation of their SPIRE SPI projects.
- Use of the assessment tools e.g. Bootcheck

11. Recommendations

The experience of the SPIRE project clearly demonstrates the feasibility of SPI in small software organisations. If you are considering an SPI project we recommend that you look at the outputs from the SPIRE project, to learn from the experience, and to ensure that you focus your project to have the most benefits for your organisation.

12. References:

Authors	References
[The SPIRE Partners]	**The SPIRE Handbook (2000)** – Better Faster, Cheaper Software Development in small organisations - ISBN 1-874303-03-7 copies available from Arcs, Seibersdorf or the SPIRE web site: http://www.cse.dcu.ie/spire
[The SPIRE Partners]	**The SPIRE Case Studies** – available from the SPIRE web site (see above)
[ISO]	**Software Process Improvement & Capability dEtermination (SPICE)** - ISO/IEC TR15504 1998
[Erwin Schoitsch]	**A Standardised Process Improvement Approach for the Development of Dependable Software-Intensive Systems.** – European Software Day, Sept. 7, 2000. – Proceedings of „European Software Day, Workshop at EUROMICRO 2000 (G. Chroust, P. Grünbacher, Eds.), p. 59-79. OCG Wien, 2000, ISBN 3-85403-200-5.
[Erwin Schoitsch, Christian Steinmann]	**Software Process Improvement – a major step to system dependability and business success.** – Proceedings of the 3rd International Austrian – Israeli Technion Symposium, Hagenberg/Linz, 26-27 April 1999.
[Bootstrap Institute]	**Bootcheck Tool** – http://www.bootstrap-institute .com
[Haase, Mülleitner & Steinmann GmbH]	**Synquest Tool** – http://www.hms.org/

QWE2000 Session 7I

Mr. Fernando T. Itakura, Ms. Silvia R. Verfilio
[Brazil]
(Crosskeys Systems Corporation)

"Automatic Support For Usability Evaluation Of A Web Application"

Key Points

- Review of different usability evaluation methods, mainly the methods implemented by TOWABE
- Reports on tools that also support usability evaluation
- Description of TOWABE functionalities and examples of its use

Presentation Abstract

Interest in World Wide Web in connection with human-computer interaction (HCI) has increased considerably over the last few years. Today there is a consense that the quality of the global system is linked with the quality of his interface. So, the development of quality interative systems demand HCI specific techniques. A key concept in HCI field is usability, which is concerned with making systems efficient, effective and easy to use. Usability evaluations are important to guarantee the global quality of a Web application. There are several methods which have proved to be useful in usability evaluation of traditional application. However, the environment in which web aplications are developed and tested are not the same of tradicitional aplications. Web applications can not be treated in the same way as a traditional application because they have some peculiar characteristics. One of them is that the development of a Web application is faster. In this sense, tools which support an automatic usability evaluation has gained importance.

TOWABE, Tool for Web Application usaBility Evaluation, is a research tool with the primary goal of automating and supporting Web usality evaluations. TOWABE is based on three well known techniques: questionnaire, card sorting and expert inspection. The tool has three modules: TCheck, TQuest and TCat. After a session using the tool, reports can be automatically generated and some statistical models are obtained from the collected data. This data can be used to compare results from differents evaluation methods. TOWABE also offers a glossary and personalization mechanisms.

About the Speaker

Fernando Takashi Itakura received his BS in Computer Science from UEL, Brazil in 1999 and is currently a masther student at Federal University of Paraná. He is

member of CE-21: 101.08 - SC21:10 of ABNT. Since February he has been teaching at UNIBEM. His research interests includes software quality, usability, software testing techniques and Internet.

Silvia Regina Vergilio received the the MS (1991) and DS (1997) degrees from University of Campinas, UNICAMP, Brazil. She is currently at the Computer Science Department at the Federal University of Parana, where she has been a faculty member since 1993. Her research interests are in the areas of Software Engineering: Program Testing, Softare Validation and Verification and Software Development Environments.

Automatic Support for Usability Evaluation of a Web Application

Fernando Takashi Itakura and Silvia Regina Vergilio

Federal University of Paraná – UFPR
Brazil

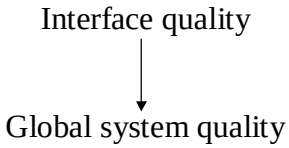
Overview of Presentation

- Introduction
- Usability Evaluation Methods
- Usability Evalution Tools
- TOWABE
- Conclusions

Introduction

- What is usability?
- ISO 9241/11 definition

  "Extent to which a product can be used by specified users to achieved their goals with effectiveness, efficiency and satisfaction in a specified context of use"



- Usability and World Wide Web
- Usability Evaluations

Goals of Work

- Review of usability evaluation methods;
- Review of tools that support usability evaluations;
- Presentation of a tool that supports and automates usability evaluations;

Usability Evaluation Methods

- Heuristic Evaluation
- Questionnaire
- Usability Inspection via Checklist
- Focus Group
- Card Sorting

Heuristic Evaluation

- Evaluators examines the interface and judge its compliance with recognized usability principles (Heuristics).
- What is the adequate number of evaluator?
  - Nielsen recommend three to five.
- I don’t like it.
- Evaluators must explain why they don’t like it.
  - It is important to indicate the respective heuristic.

Questionnaire

- a set of questions requiring a response which describes past behaviors, the user expectations, attitudes and opinion.
- Important considerations:
  - Designing a questionnaire is not writing a book.
  - Questionnaire should be checked before answered in the real test.
  - Do not put many open-ended questions.
  - Use discrete scales.

Usability Inspection using Checklist

- Review of a system based on a set of guidelines.

the quality of the tool (checklist)

the quality of the evaluation.

- Potentialities:
  - Anyone can perform the checklist;
  - Systematization of the evaluation;
  - Reduction of subjectiveness;
  - Identification of usability problems frequently found.

Focus Group

- Moderator
- Representative users
- Stimulus

Card Sorting

- A categorization method.

- The proccess consist:
  - List of all items that will be sorted;
  - Write this list in cards;
  - Give these cards to users;
  - The users classify these items in groups;
  - The users label the groups;
  - A categorization model is obtained.

Usability Evalution Tools

- QUIS
- Ergolist
- IBM UCD WORKBENCH
- BOBBY
- ERGOLIGHT USABILITY TOOLS

TOWABE - Motivations

- Most tools supports only one method;
- The evaluations methods are complementary;
- Comparasion among methods are desired;
- Combination of a formal, well-known and reliable checklist (ISO 9241)- but not focused on World Wide Web - and a checklist focused on World Wide Web.

TOWABE

- TOWABE - **TO**ol for **W**eb **A**pplication usa**B**ility **E**valuation.
- It is based on three well-known techniques: questionnaire, card sorting and usability inspections using checklist.
- It has three modules:
  - TCheck
  - TQuest
  - TCat
- Two different groups:
  - Evaluator group.
  - General User group.

TOWABE - cont

- Passwords are provided to guarantee the distinction among the groups and the integrity of the evaluation session.
- It is possible to obtain statistical models.
- It is possible to compare the results from different evaluation session.
- Personalization mechanisms is avaliable.
- TOWABE identify the user characteristics.

TCheck module

- TCheck module provides a checklist to assess the usability of Web applications.
- The checklist was developed based on:
  - ISO 9241.
  - BIR/1198.

	YES	NO	N.A.
1. Links and buttons are recognizable.			
2. Help information is task-dependent.			

TQuest module

- The TQuest module implements a questionnaire projected to assess the users' satisfaction.
- standard questionnaire is avaliable
  - The Evaluator group is able to construct his own personalized questionnaire.

	Completely agree ◄ .... ► Completely disagree	No option
1. Remember where I am in the Website is very difficult. **Comments:**	o o o o o	☐
2. The terminology used in the software reflects that of my work environment. **Comments:**	o o o o o	☐

TCat module

- It is based on Card Sorting method.
- It allows the Evaluator group to determine how well the categories and items of the interface are understood by the General Users group.

Conclusions

- TOWABE implements more than one method.
  - This permits to compare the results obtained from different methods.
- A checklist that combines a formal, well-known and reliable checklist (ISO 9241), but not focused on World Wide Web with other checklist focused on World Wide Web BIR/1198 was obtained (TCheck checklist).
- Personalization mechanisms is available.
- Statistical models can be obtained automatically.

Automatic Support for Usability Evaluation of a Web Application

Fernando Takashi Itakura
fitakura@onda.com.br

Silvia Regina Vergilio
silvia@inf.ufpr.br

Computer Science Department - C.P. 1601
Federal University of Paraná – UFPR
Centro Politécnico – Jardim das Américas
Curitiba- PR, Brazil.
CEP: 19081-970

Abstract

The requirements for quality of Word Wide Web applications have increased considerably in the last years. In this context, a key concept to ensure the quality of software interfaces is usability. Different usability evaluation methods are found in the literature and tools, that support automatic usability evaluations based on these methods, have gained importance. However most tools are based on only one method and comparisons among different methods are not very easy. This work presents a tool for automatic usability evaluation, named TOWABE, based on three well-known methods: questionnaire, card sorting and checklist. These methods are complementary and can reveal different usability problems. After a session using the tool, reports can be automatically generated and some statistical models are obtained from the collected data. This data can be used to design the interface and to compare results from different evaluation methods. TOWABE also offers a glossary and personalization mechanisms.

1. INTRODUCTION

The interest in World Wide Web in connection with Human-Computer Interaction (HCI) has increased considerably over the last few years. Today there is a consense that the quality of the global system is related to the quality of its interface [4]. So, the development of interactive systems demands Human-Computer Interaction specific techniques. A key concept in HCI field is usability, which is concerned about making systems efficient, effective and easy to use.

These increasing requirements for software quality have lead the community to produce several guidelines, standards and norms to guide the evaluation of a software product or process [1], [8], [18]. The ISO 9241 is one of this standard,

which emphasizes ergonomic and usability requirements [8]. ISO 9241-11 defines usability as an extent to which a product can be used by specified users to achieved their goals with effectiveness, efficiency and satisfaction in a specified context of use.

Then, user performance can be measured by the accuracy and completeness with which a user achieves specified goals (effectiveness) and by the resources spent on this task (efficiency). User satisfaction can be measured by freedom from discomfort.

ISO 9241-11 also emphasizes that usability is dependent on the context of use [8]. The context of use consists of the users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used. Then, the level of usability achieved will depend on the specific circumstances in which a system is used.

A Web application can have excellent quality of use for some people and poor quality of use for others [12]. For example, a graphical user interface may use a formal and technical language – which experts can understand and use successfully and safely – but can be very frustrating for novice users because they are not familiar with this language. In this sense usability evaluations are very important to guarantee the global quality of a Web application.

In the literature, there are several methods that have proved to be useful in usability evaluation of traditional applications. However, the environments in which web applications are developed and tested are not the same of traditional applications. Web applications can not be treated in the same way as a traditional application because they have some peculiar characteristics. One of them is that the development of a Web application is faster. When this characteristic is considered, tools that support an automatic usability evaluation are fundamental.

The goal of this work is to present a tool named TOWABE (Tool for Web Application Usability Evaluation) which supports automatic usability evaluation of applications Web based on three well known methods: questionnaires, checklists and card sorting. An evaluation session using the tool permits the comparison of results obtained from different evaluation methods and the generation of reports. TOWABE also offers a glossary and personalization mechanisms.

The paper is organized as follows. Section 2 presents a review of the different usability evaluation methods, mainly the methods supported by TOWABE. Section 3 presents similar works, reporting some tools that also support usability evaluation and that are motivations for TOWABE implementation. Section 4 describes TOWABE functionalities. Section 5 concludes the paper.

2. USABILITY EVALUATION METHODS

This section presents a review of usability evaluation methods. Using different methods to evaluate a system is very important because they can help identify different kind of usability problems. The methods presented must be viewed as complementary.

2.1. Heuristic Evaluation

Heuristic Evaluation is a method for finding the usability problems in a user interface design. During the heuristic evaluation a small set of evaluators examines the interface and judges its compliance with recognized usability principles [14].

To perform heuristic evaluation each individual evaluator inspects the application interface individually. Only after all evaluations have been completed are the evaluators allowed to communicate and have their findings aggregated. The results of the evaluation can be recorded either as written reports from each evaluator or by verbalization of their comments to an observer as they go through the interface [14]. When the evaluators present the results it is not sufficient for evaluators to simply say that they do not like something; they should explain why they do not like it with reference to the heuristics. An important issue in heuristic evaluation is the number of evaluators. In principle, individual evaluators can perform a heuristic evaluation of a user interface, but studies recommended three to five evaluators.

A lot of heuristics was proposed by different authors, but Molich and Nielsen [13] heuristics are the most known and were used in this work. Below ten heuristics, extracted from [15] are presented.

1. Visibility of system status: the system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. Match between system and the real world: the system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom: users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.

4. Consistency and standards: users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention: error messages are very important to prevent a problem from occurring in the first place.

6. Recognition rather than recall: make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use: accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design: dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors: error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation: even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

2.2. Questionnaire

A questionnaire is a set of written questions requiring a written response that describes past behaviors, the user expectations, attitudes and opinions towards the system [3].

For being really able to generalize the conclusions, some important principles should be met [3]:

- Designing a questionnaire is not writing a book. Long questionnaires are very boring for the users, difficult to analyze, and in general, not worth the cost.
- Questionnaire should be checked before answered in the real test.
- A questionnaire with many open-ended questions is not a good questionnaire.

- Multiple choice questions are suited for those issues, which are comprehensively, addressed (all the possible response alternatives are known)

- Most usual rating scales are discrete scales. A well-designed questionnaire very often produces very goods results, and can be used to compare subjective opinions over different situations, systems and development phases.

2.3. Usability Inspection using Checklist

Usability inspection using checklist is review of a system based on a set of guidelines. In this method, the quality of the tool (checklist) will determine the potentialities of the evaluation. Checklist well elaborated must produces uniform and wide results [4].

The evaluation using checklists presents the following potentialities [4]:

- It can be performed by system developers, it is not necessary a usability expert. The knowledge is intrinsic to the checklist;

- Systematization of the evaluation;

- It facilitates the identification of usability problem focused by the checklist;

- It increases the effectiveness, once there is a subjective reduction;

- It decreases the evaluation cost because is a fast method and does not require a usability expert.

This method offers the identification of a large number of small usability problems that is frequently found in system interfaces [10].

2.4. Focus Group

Focus group is a data collecting technique where representative users – normally 6 to 12 - are brought together to discuss issues relating to the system. A usability engineer play the role of a moderator, who needs to prepare the list of issues to be discussed beforehand and seek to gather the needed information from the discussion [3].

The determination of the number of participants needs some precaution. First, never select large groups, they are very difficult to manage and the users' individual participation is poor. There is evidence that group size is inversely related to the degree of participation.

This method depends essentially on the moderator style and skills. The moderator needs to be skilled in group facilitation and communication to make a focus group successful.

One interesting characteristic of focus group method is that the opinions about a variety of issues are gerenarally determined not by individual information gathering but through communication with others.

2.5. Card Sorting

Card Sorting is not a usability evaluation method, but is a categorization method where users sort cards depicting various concepts into several categories [6].

The card sorting method starts writing a list of all items that will be sorted and writing each item on a separate index card. After, the users receive the cards, classify them into groups and finally label the groups created. The diverse groups form a user categorization model to the system interface. The interface can be designed according to this model.

3. RELATED WORKS

This section describes some usability evaluation tools that implement some of the methods from last section. The TOWABE, presented in this work, is strongly based on these methods and tools.

3.1. QUIS – Questionnaire for User Interaction Satisfaction

The Questionnaire for User Interaction Satisfaction (QUIS) is a tool developed at the University of Maryland at College Park [16]. The QUIS was designed to assess user's subjective satisfaction with specific aspects of a human-computer interface. The current version contains a demographic questionnaire, a measure of overall system satisfaction, and hierarchically organized measures of eleven specific interface factors such as screen factors, terminology and system feedback, learning

factors, system capabilities, technical manuals, on-line tutorials, multimedia, voice recognition, virtual environments, internet access and software installation.

3.2. ERGOLIST

The Ergolist was developed by Labiutil – Usability Laboratory – at the Federal University of Santa Catarina, Brazil [11] The Ergolist is a tool designed to evaluate interactive software using Internet. Three modules compose the Ergolist. The first module performs a systematic inspection of the interface quality, using eighteen checklists that are based on specific ergonomic criteria. The second one shows in an informal way how the checklist is composed. The last one presents ergonomics recommendations that can be useful in the interface project.

3.3. IBM UCD WORKBENCH

IBM UCD WORKBENCH is a set of tools for User-Centered Design in the development of products and applications. Currently, two tools compose the set: UCD Satisfaction Survey and UCD Questionnaire Resources [7]. UCD Satisfaction Survey uses the Web to gather requirements and satisfaction information from users. The tool consists of an application that automatically generates layout, scripting, and code for online surveys. It also collects and tabulates results dynamically on the Web. UCD Questionnaire Resources provides a set of Web survey templates in the Net Objects Fusion authoring environment. The templates are for determining customer satisfaction, eliciting product requirements, identifying user tasks and use scenarios, determining current feature usage, and evaluating icons and terminology.

There are in the literature other tools implementing the methods mentioned in Section 2 [2], [5]. However most existent tools supports only one method. This becomes the task of comparing information about evaluations using different methods very hard. The TOWABE, described in next section, supports different evaluation methods and considers their complementary aspects.

4. TOWABE

TOWABE – TOol for Web Application usaBility Evaluation – is a research tool with the primary goal of automating and supporting Web applications usability evaluations. TOWABE is based on three well-known techniques: questionnaire, card sorting and expert inspection. The tool has three modules: TCheck, TQuest and TCat. After a session using the tool, reports can be automatically generated and some statistical models are obtained from the collected data. This data can be used to

compare results from different evaluation methods. TOWABE also offers a glossary and personalization mechanisms.

Two different users group uses TOWABE: Evaluator and General User. The first group is composed by experts in evaluation of web applications or by the developers of the application being evaluated. The Evaluator group creates an evaluation session and generally invites people to compose the second group. The second group uses the software and collaborates with the evaluation session. Therefore, sometimes a person, in a specific situation, will belong to Evaluator group, and in other situation will belong to General User group. When an evaluator creates an evaluation session the tool provides two passwords. The first will be used by Evaluator group to enter again in the created session, and the second will provide access to the General Users group.

4.1. TCheck module

The TCheck module provides a checklist to assess the usability of Web applications. A glossary is also avaliable to easy the understanding of doubts that can appear during checklist utilization.

The checklist implemented by TCheck was developed based on ISO 9241 [8] and BIR/1198 [1]. One important characteristic obtained from this combination is a checklist that combines a formal, well-known and reliable checklist (ISO 9241), but not focused on World Wide Web and a checklist focused on World Wide Web. The complete checklist can be found in [9]. Figure 1 shows an example of a checklist item. For each item there is three answer options: YES, NO, N.A. (not applicable).

The preparation of the evaluation report is supported by TCheck module. All results of the evaluation process are saved to a text file where all the recommendations are listed. When a NO option is checked TOWABE automatically put the respective recommendation into text file.

	YES	NO	N.A.
1. Links and buttons are recognizable.			
2. Help information is task-dependent.			

Figure 1 – Example of a checklist item.

4.2. TQuest module

The TQuest module implements a questionnaire, projected to assess the users' satisfaction with specific aspects of the application interface being evaluated.

A standard questionnaire is avaliable, however the evaluator in the Evaluator group is able to construct or use his own personalized questionnaire, some questions can be enabled or not depending on the characteristics of the application, etc. Of course, that this option must be used by very experienced evaluator in due to the risk of using a no reliable questionnaire.

This module also generates two passwords, one for Evaluator group and other for the General User Group. Evaluator password permits to get the results obtained from evaluation. The General User Group password permits to modify the questionnaire created by the evaluator.

Another important aspect is that TOWABE identifies the user invited by the evaluator to participate of an evaluation session. Relevant characteristics as knowledge, skill, experience, education, training, physical attributes and nationality are report. So, statistical models can allow the evaluator to know the results from all users or from a specific group with certain characteristics.

The TQuest questions are answered with 5-point scale. And there is a plus option – No option – when the user for some reason do not wish to express or do not have an option. It is also possible make some comments. The answer and the comments are stored by TOWABE to generate the reports required by Evaluator group. Figure 2 shows an example of two TQuest questions.

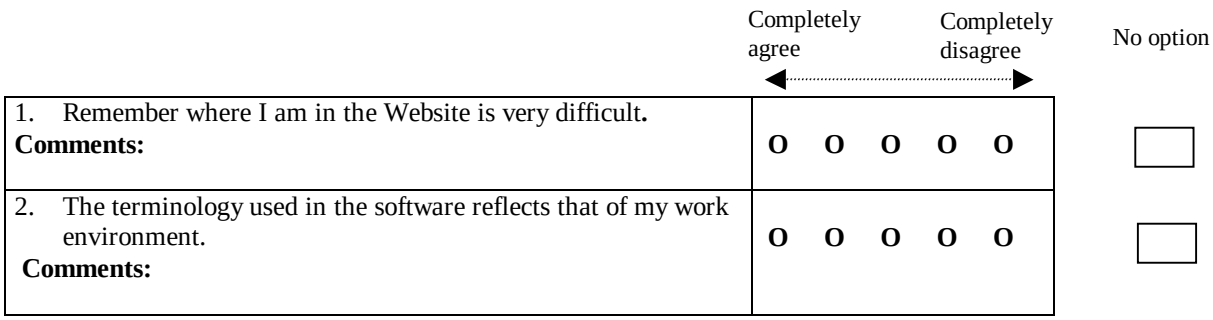


Figure 2 – TQuest question example

4.3. TCat module

The TCat module allows the Evaluator group to determine how well the categories and items of the interface are understood by the General Users group. Initially the evaluator provides the session name, the categories and the respective interface items. With this information, TOWABE creates a baseline that will be used, in the future, and will be compared with the user categorization model. This model is obtained from the General Users group, during an evaluation session using TCat.

In this moment, TOWABE also generates and supplies the two passwords mentioned. Hence, the session is created and then the evaluator can invite the users to utilize this specific session. The evaluator can obtain comparative data automatically generated by the tool. Statistical data like number of users who participated of the evaluation session, number of users who put the item in the same category of the baseline and other statistical measures as standard deviation is given.

In TCat module when the evaluator invite a person to participate of an evaluation session, TOWABE also identifies this user in the same way as the TQuest Module.

5. CONCLUSIONS

This work presented a review of usability evaluation methods that can be applied for usability evaluation of a Web application. Some tools that support these methods were also presented. Most tools implement only one method. However, using different methods to evaluate a system is very important because they can help identify different kinds of usability problems. Considering these issues, a research tool – TOWABE – is proposed with the primary goal of automating and supporting Web different usability evaluation methods: questionnaire, checklist and card sorting. This TOWABE characteristic permits to compare the results obtained from these different methods.

The checklist implemented by the module TCheck is based on ISO 9241 [8] and BIR/1198 [1]. One important characteristic obtained from this combination is a checklist that combines a formal, well-known and reliable checklist (ISO 9241), but not focused on World Wide Web with other checklist focused on World Wide Web. Besides of this, personalization's mechanisms are available. This allows the users (the evaluators) to create their own questionnaires considering for example specific issues of the application being evaluated.

After a session, TOWABE users also obtain recommendations that can improve the usability of the Web application; they can also get statistical models that will help in the development of a Web application.

TOWABE is being implemented and a case study to evaluate data obtained from TOWABE utilization is being planned.

REFERENCES

[1] BIR/1198 – Guia para Projetos de Aplicação para Web*. Bhase Internal Report,* November, 1998, Londrina, PR- Brazil. (In Portuguese)

[2] Bobby. http://www.cast.org/bobby/index.html

[3] CEREZO, P. et al. *USINACTS Tutorial.* http://at.hhi.de/USINACTS/tutorial/intro.html.

[4] CYBIS, W. et al. *Avaliação de Usabilidade de Software de Escritório: Uma perspectiva ergonômica.* Proc. II Seminário Desenv. Software de Acordo com Padrões Internacionais de Qualidade e Produtividade, Curitiba, 2000. (in Portuguese)

[5] ErgoLight Usability Tools http://www.ergolight-sw.com/

[6] HOM, J. *The Usability Methods Toolbox.* http://www.best.com/~jthom/usability/usable.htm

[7] IBM. E-business tools. http://www.ibm.com/ibm/hci

[8] ISO 9241: Ergonomic requirements for office work with visual display terminals (VDTs), 1996.

[9] ITAKURA, F. *Uma Ferramenta para Avaliação de Aplicações para Web. Master Dissertation (in elaboration)*, Curitiba, 2000. (In Portuguese)

[10] JEFFREIES, R. et al. *User Interface Evaluation in the Real World: A Comparison of Four Techniques.* In Proc. Conf. Human Factors i Computing Systems, CHI'91, pp 119-124, ACM Press, 1991.

[11] LABIUTIL. *Laboratório de Utilizabilidade.* http://www.labiutil.inf.ufsc.br (in Portuguese)

[12] MACLEOD, M. *Usability: Pratical Methods for Testing and Improvement.* Proceedings of the Norwegian Computer Society Software '94 Conference. Sandvika, Norway, 1- 4, 1994.

[13] Molich, R., and Nielsen, J. (1990). Improving a human-computer dialogue, *Communications of the ACM* **33**, 3 (March), 338-348.

[14] NIELSEN, J *How to Conduct a Heuristic Evaluation.* http://www.useit.com/papers/heuristic/heuristic_evaluation.html

[15] NIELSEN, J. *Ten Usability Heuristics.* http://www.useit.com/papers/heuristic/heuristic_list.html

[16] QUIS. *Questionnaire for User Interaction Satisfaction.* http://www.lap.umd.edu/QUIS/index.html.

[17] SMITH & MOSIER. *Guidelines for Designig User Interface Software.*

[18] STEIGER, P. Checklist to Assess Websites according to Usability Criteria. http://www.swisschi.ch

QWE2000 Session 7M



Mr. Luis Filipe D. Machado, Ms. Kathia M. de Oliveira & Ms. Ana Regina C. Rocha [Brazil] (Federal University Of Rio de Janeiro)

"Using Standards And Maturity Models For The Software Process Definition"

Key Points

- Software Process
- Software Quality
- Maturity Models

Presentation Abstract

During the last years, the software product has increased in size and complexity, assuming a critic and strategic role in the organizationsÝ business. In this scenario, obtaining software products with quality, under the time limits and with the resources allocated to the projects, became a challenge. Software process definition is a fundamental requirement to guarantee the quality of software products. Nevertheless, the effectiveness of such processes depends on their adequacy to the characteristics of the organization and of the product whose development is desired, as well as of the project. In an organization, different processes can coexist adequately with different projects. To organize and discipline the software development it is important to determine the fundamental activities that shall be present in any defined process. Consequently, the definition of a standard process establishes a common structure to be used by the organization in its software projects, as it sets the basis for the definition of all processes. With basis on the standard process one can define software processes for different projects. To

develop this idea we have set up two main requirements: (i) the combined use of ISO 12207 and maturity models (such as SPICE and CMM) and (ii) a three-step approach that includes the definition of a standard process, its specialization for different approaches of development and its instantiation for each specific project.

About the Speaker

Luis Filipe D. Cavalcanti Machado (filipe@cos.ufrj.br) received his MSc degree in Computer Science from the Federal University of Rio de Janeiro in 2000. Now he is a PhD student at the same institution. He is the author of 2 articles presented at international congresses.

K¤thia Mar‡al de Oliveira (kathia@cos.ufrj.br) received her PhD degree in Computer Science from the Federal University of Rio de Janeiro in 1999. She is research associate at the Federal University of Rio de Janeiro. She is the author of 20 articles presented at international congresses.

Ana Regina C. Rocha (darocha@cos.ufrj.br) received her PhD degree in Computer Science from the Federal University of Rio de Janeiro. At present she is an associate professor at the Graduate School of Engineering of the Federal University of Rio de Janeiro. She is the author of 2 books and of more than 200 articles, which have appeared in specialized magazines and/or were presented at national and international congresses.

Using Standards and Maturity Models for the Software Process Definition

Luis Filipe D. Cavalcanti Machado

Káthia Marçal de Oliveira

Ana Regina Cavalcanti da Rocha

{filipe, kathia, darocha}@cos.ufrj.br

COPPE -UFRJ

QWE - 2000
Brussels, BELGIUM

Introduction

- **Software engineering concerns**
  - ⇨ Product quality
  - ⇨ Development process quality
- **The number of defects present in software delivered for testing is a direct function of the quality of the process used to construct the software**
  - ⇨ Tests can only detect 70% of the latent defects in the code
  - ⇨ Inspections can detect 80 - 90% of the errors prior to testing

But,

a good process avoids the presence of defects in the product

Process Quality

Approaches:

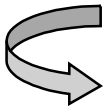
- ⇨ **ISO 12207**
- ⇨ **ISO 9000-3**
- ⇨ **CMM**
- ⇨ **ISO/IEC TR 15504**
- ⇨ **BOOTSTRAP**
- ⇨ **TRILLIUM**

Usual Problems in Software Companies

- **Lack of knowledge about ISO 9000-3, CMM and SPICE by managers and developers**
- **Unsystematized and outdated Engineering Software Training**
- **No defined process – ad-hoc or managed development**
- **Difficulties in introducing new technologies**
- **Difficulty in managing projects**
- **Managers spend most of their time “putting out fires”**
- **Unsatisfied managers and developers**

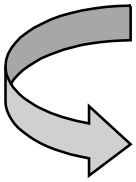
The “maturity movement”

- **Managers are starting to recognize the need for a software process defined for the company and followed by all**
- **Internationalization of the software industry ➨ Need to define these processes adherent to international standards**
- **The definition, use and continued improvement of the software process is becoming the main objective for organizations**



- **Main difficulties:**
  - ⇨ Standards and models with different objectives and characteristics
  - ⇨ Understanding the many aspects of software engineering
  - ⇨ Lack of a generically applied software process

Software Process



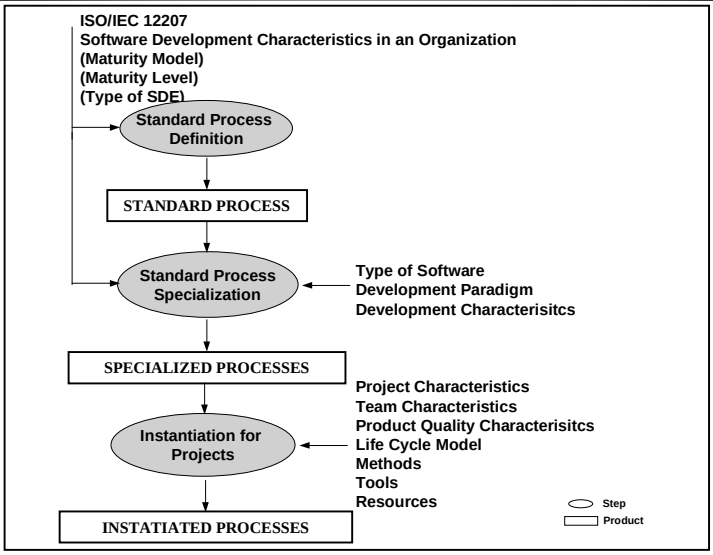
Depends on

- ⇨ **Type of systems**
- ⇨ **Application domains**
- ⇨ **Organizations and teams**
- ⇨ **Business constraints (schedule, cost, quality)**

Objective

- **Present a three-step model for the definition of software processes, based on international standards, organization and specific projects characteristics.**
- **Present a tool supporting the definition of the standard process for an organization, considering the ISO/IEC 12207 Standard and the maturity models (CMM / ISO-IEC TR 15504).**

Model for Software Processes Definition



Model for Software Process Definition

■ **Define and organize some of the main factors influencing the definition of a software process**

■ **Steps:**

- ⇨ Definition of the standard process
- ⇨ Specialization of the standard process
- ⇨ Instantiation for specific projects

■ **The ISO/IEC 12207 Standard is used as a basis for defining any software process**

Definition of the Standard Process

■ **The basic process guiding the establishment of a common process in the organization, describing the fundamental elements which it is hoped are incorporated in any defined process.**

■ **Set of software processes relevant to the context of the organization.**

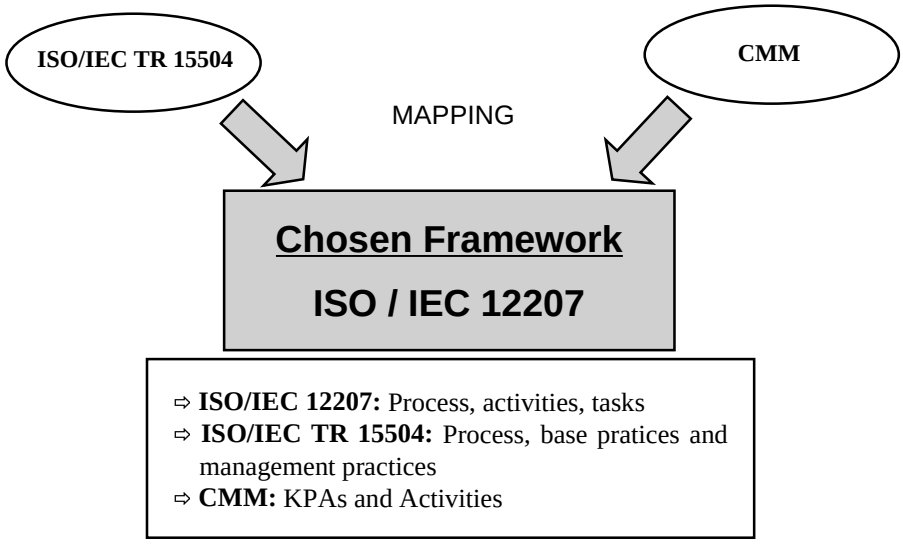
■ **Factors influencing the definition:**

- ⇨ The ISO/IEC 12207 Standard
- ⇨ Software Development Characteristics in the Organization
- ⇨ Maturity Model (ISO/IEC TR 15504 and CMM)
- ⇨ Maturity Level
- ⇨ Type of SDE

Definition of the Standard Process
(The ISO/IEC 12207 Standard and the Maturity Models)

- **The ISO/IEC 12207 Standard and the maturity models define activities which shall be incorporated in a software process.**
- **To enable the combined use of these standards, a mapping was set up, among the ISO/IEC 12207 Standard and the maturity models (ISO/IEC TR 15504 and CMM).**
- **The mapping allows the software engineer to analyze each activity to be defined, comparing it with one or more equivalent activities of the ISO/IEC 12207 Standard.**

Combined use of ISO/IEC 12207 Standard with the Maturity Models



Configuration Management Process

ISO/IEC 12207 Activities	ISO/IEC TR 15504 Base Practices (BP)	CMM 1.1 KPAs and Activities
• Process implementation (6.2.1).	• Develop configuration management strategy (SUP.2.BP1).	• KPA Software Configuration Management Activity 1. • KPA Software Configuration Management Activity 2.
—	• Establish configuration management system (SUP.2.BP2).	• KPA Software Configuration Management Activity 3.
• Configuration identification (6.2.2).	• Identify configuration items (SUP.2.BP3). • Maintain configuration item description (SUP.2.BP4).	• KPA Software Configuration Management Activity 4.
• Configuration control (6.2.3).	• Manage changes (SUP. 2.BP5). • Manage product releases (SUP. 2.BP6). • Maintain configuration item history (SUP.2.BP7).	• KPA Software Configuration Management Activity 5. • KPA Software Configuration Management Activity 6. • KPA Software Configuration Management Activity 8.
• Configuration status accounting (6.2.4.)	• Report configuration status (SUP. 2.BP8).	• KPA Software Configuration Management Activity 9.
• Configuration evaluation (6.2.5).	—	• KPA Software Configuration Management Activity 10.
• Release management and delivery (6.2.6).	• Manage the release and delivery of configuration items (SUP. 2.BP9).	• KPA Software Configuration Management Activity 7.

QWE 2000 - Luis Filipe Cavalcanti Machado

Software Life Cycle Processes

Primary Processes	Supporting Processes	Organization Processes
• Acquisition (ISC) • Supply (IS) • Requirements elicitation (SC) • Development (ISC) • Maintenance (IS) • Operation (IS)	• Documentation (ISC) • Configuration management (ISC) • Quality assurance (ISC) • Verification (ISC) • Validation (ISC) • Joint review (ISC) • Audit (ISC) • Problem resolution (ISC) • Defects prevention (C)	• Management (ISC) • Infrastructure (ISC) • Improvement (ISC) • Training (IC) • Project management (S) • Quality management (S) • Risk management (S) • Organizational alignment (S) • Human resource management (S) • Measurement (S) • Reuse (S) • Process change management (C) • Technology change management (C) • Quantitative process management (C) • Intergroup coordination (C)

Observations:
(I) Process defined by ISO/IEC 12207; (S) Process defined by SPICE (ISO/IEC TR 15504)
(C) Process derived from KPA of CMM.

QWE 2000 - Luis Filipe Cavalcanti Machado

Definition of the Standard Process
(Software Development Characteristics in the Organization)

- **Incorporate characteristics peculiar to each organization, in the defined processes**
- **Related characteristics**
  - ⇨ to the work environment
  - ⇨ to the knowledge and experience of the teams involved
  - ⇨ to the culture and experience of the organization in developing the software

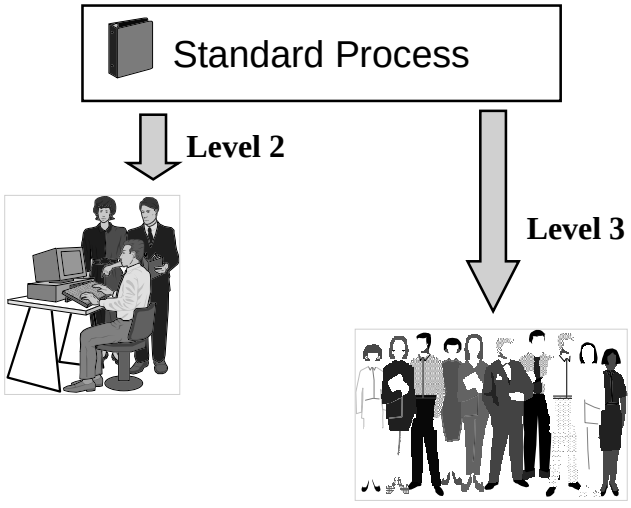
Specialization of the Standard Process

- **Factors influencing specialization**
  - ⇨ type of software (Expert systems, information systems,…)
  - ⇨ development paradigm (OO or structured)
  - ⇨ maturity model / level (teams with different capability levels in software engineering)
  - ⇨ type of SDE (Software Development Environment)
- **Specific activities may be added or modified in accordance with the context for which the specialization is being carried out**

Specialization of the Standard Process
(Type of Software)

- **Interactive Software:**
  - ⇨ Optimize the human-machine interaction, including commands and functions which improve response times
  - ⇨ Optimize aspects of interface and interactions, including data types, text size, graphics and colors use
- **Different software types require different testing strategies:**
  - ⇨ Expert systems test
  - ⇨ Object Oriented systems test
  - ⇨ ...

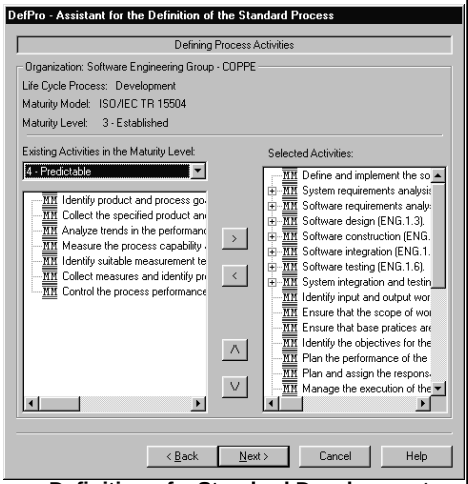
Specialization of the Standard Process
(Maturity Model/Level))



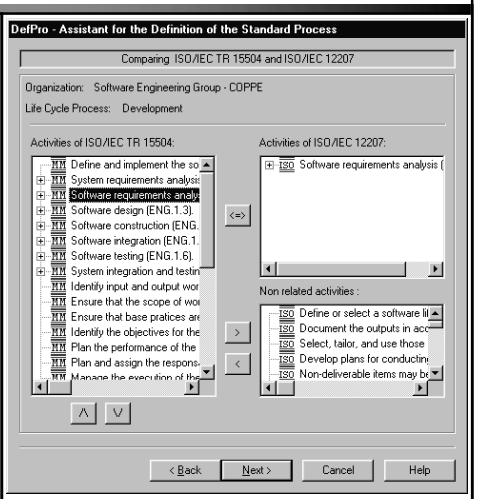
Instantiation for Specific Projects

- **Factors influencing instantiation:**
  - ⇨ **Project characteristics** (timetable, cost)
  - ⇨ **Team characteristics** (software development experience, application domain experience)
  - ⇨ **Life cycle model**
  - ⇨ **Product quality characteristics** (established based on quality criteria provided in ISO/IEC 9126 and on interviews with the users)
  - ⇨ **Methods, tools and resources**

Def-Pro Tool

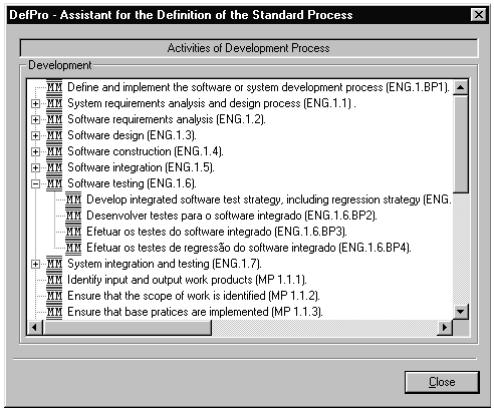


Definition of a Standard Development Process adherent to Level 3 of the ISO/IEC TR 15504

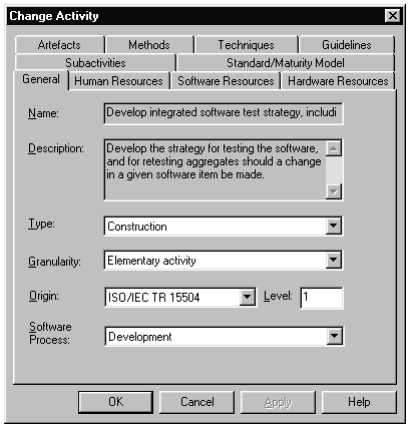


Comparative analysis between the activities of ISO/IEC TR 15504 and ISO/IEC 12207

Def-Pro Tool



Development Standard Process Activities



Activity Definition

Conclusion

- **The use of international standards in the area of software processes has become an important factor for organizations entering the so-called "maturity movement".**
- **The combined use of the Standard and maturity models, however requires an understanding of its differences and similarities, which implies difficulties in its use. Mapping among these standards permits their combined use in the definition of software processes.**
- **The proposed model considers the definition, specialization and instantiation of software processes, allowing definition of basic activities which will make up the standard process, as well as the definition of activities adequate for the different development contexts and capability levels in Software Engineering.**
- **The`importance of knowledge acquired and modeled on software processes and international standards, and the automated support assisting the software engineer in the definition of the software processes.**

USING STANDARDS AND MATURITY MODELS FOR THE SOFTWARE PROCESS DEFINITION

Luis Filipe D. Cavalcanti Machado, Káthia M. de Oliveira, Ana Regina C. Rocha

Federal University of Rio de Janeiro, Graduate School of Engineering,
Computer Science Department, Caixa Postal 68511 – CEP 21945-970
Rio de Janeiro, Rio de Janeiro, Brazil
e-mail: filipe@cos.ufrj.br; kathia@cos.ufrj.br; darocha@cos.ufrj.br

ABSTRACT

During the last years, the software product has increased in size and complexity, assuming a critic and strategic role in the organizations' business. In this scenario, the obtainment of software products with quality, under the time limits and resources established in the projects, became a challenge. Software process definition is a fundamental requirement to guarantee the quality of software products. Nevertheless, the effectiveness of such processes depends on their adequacy to the characteristics of the organization, of the product desired to be developed, and of the project. In an organization, various processes can coexist adequate to different projects. To organize and discipline the software development it is important to determine the fundamental activities that shall be present in any defined process. Consequently, the definition of a standard process establishes a common structure to be used by the organization in its software projects, as it institutes the basis for the definition of all processes. To develop this idea we have set up two main requirements: (i) the combined use of ISO 12207 and maturity models (such as SPICE and CMM) and (ii) a three-step approach that includes the definition of a standard process, its specialization for different approaches of development and its instantiation for each specific project.

Key-Words: Software Process, Software Quality, Maturity Models.

1. Introduction

Over the last few years, the software product has increased in size and complexity. Organizations, in search of competitive advantages, have invested heavily in automating its business processes. Greater responsibilities have been attributed to the software product, to the point that software has assumed a critical and strategic role in organizations business. In this scenario, it has become an increasing challenge to carry out software projects within the specified time and with the scheduled resources. (ZAHRAN, 1998).

The experience of the software industry shows that when projects are unsuccessful, the main reason lies in the lack of a disciplined software process, or that is, there is no mechanism to enable product quality management and control. Following this same tendency, it is already widely accepted that the quality of a software product is strongly determined by the quality of the process used in its development and maintenance (PFLEEGER, 1998).

Although the importance of evaluations and continuous improvements in the processes are currently widely discussed, the software engineers and organizations still have difficulty in defining processes, basically because a single generically applicable software process does not exist (JACOBSON *et al.*, 1998; ROCHA *et al.*, 1999). The processes vary because there are different types of systems, application domains, teams, organizations, and as well each has its own business restrictions, such as timetable, cost, quality and reliability (JACOBSON *et al.*, 1998). The definition of the software process is an activity requiring experience and involves the knowledge of many aspects of software engineering. The combined use of standards and maturity models contained in the literature, is an important factor when defining a process. Its use,

however, is not a simple activity, requiring a good understanding of standards, and as well the characteristics appropriate to project for which the process is being defined.

With the advent of international standards and the highest preoccupation with software quality, organizations adopt the ISO/IEC 12207 (1995) Standard and the maturity models, such as CMM, TRILLIUM, BOOTSTRAP and SPICE (ZAHRAN, 1998; EMAM *et al.*, 1998; PAULK *et al.*, 1997; ISO/IEC TR 15504, 1998) in the definition of software processes. In this context, organizations have used maturity models as a guide to examine their software development practices, and are continually improving them in order to reach the requirements imposed by the ISO/IEC 12207 Standard (FERGUNSON and SHEARD, 1998). Capability-level structuring of the maturity models allows objectives to be set, and reached as a function of the improved capacity on the part of the organization to improve its quest to reach the requirements stipulated for the ISO/IEC 12207 Standard. Experiences in the literature have proven the success of this approach, with the combined use of the ISO/IEC 12207 Standard and CMM (FERGUNSON and SHEARD, 1998; MACHADO *et al.*, 1999).

This article describes an approach to defining software processes based on the combined use of the ISO/IEC 12207 Standard and maturity models, specifically CMM together with the ISO/IEC TR 15504. This approach was applied in a model for the definition, specialization and instantiation of software processes, with the objective of allowing the definition of processes compatible with international standards and models. Section 2 presents an approach which allows the combined use of the ISO/IEC 12207 Standard together with maturity models. Section 3 describes how the referred approach was incorporated into a model allowing software processes to be defined, in line with the characteristics of the organizations and specifically, the projects, and as well discusses some of the main factors involved with defining the software processes. Section 4 details how the referred approach has been applied to a tool to assist the software engineer in defining a standard process for an organization. Conclusions are then presented in section 5.

2. Approach for the combined use of the ISO/IEC 12207 Standard with the Maturity Models

The use of maturity models has been an interesting strategy, allowing organizations to gradually reach the requirements imposed by the ISO/IEC 12207 Standard (FERGUNSON and SHEARD, 1998; MAIDANTCHIK *et al.*, 1999; MACHADO *et. al.*, 1999). The referred Standard defines a set of activities to be carried out by organizations involved in the acquisition, development, operation, maintenance and supply of software products; it however does not propose an approach for organizations to improve their software processes, and to adhere to it. The maturity models, structured by capability levels, would be a natural path for organizations looking to establish action priorities with the view to improve their processes.

The purpose of ISO/IEC 12207 was to establish a common structure for the definition of software processes. Both CMM and ISO/IEC TR 15504 were developed to evaluate processes and determine their capability, showing differences in the way the two models are used while reaching this objective (ROUT, 1996; ROUT, 1998). Although with different objectives, either ISO/IEC 12207 or ISO/IEC TR 15504 and CMM describe activities that need to be incorporated into a software process. These standards describe such activities using different denominations and detail levels, which make difficult their combined used in the process definition.

Therefore, in order for the process definition to adhere to ISO/IEC 12207 and to the maturity models, an approach has been established for mapping the processes, key process areas, activities and tasks included in the ISO Standard and in the Models. In the proposed approach,

ISO/IEC 12207 with its structure divided into life cycle processes, was used as a basis for the definition of any software process, and a group of procedures was adopted to enable conformity between the structure of these models and the referred Standard. A high-level mapping among the standards had already been proposed in prior articles (ISO/IEC TR 15504, 1998; SEI, 1998), but not at a sufficient level of detail to permit organizations to define their software processes. The approach in this article proposes to establish a relationship among the international standards, and also to permit this relationship to be used for automated support when defining the software processes, as presented in section 4. Sections 2.1 and 2.2 below outline the procedures used to permit mapping between the activities contained in the maturity models and in the ISO/IEC 12207 Standard. Section 2.3 presents the life cycle processes resulting from the proposed approach, and as well an example of mapping between the activities of the ISO/IEC 12207 Standard and the maturity models.

2.1 Mapping the Future ISO/IEC 15504 Standard with the ISO/IEC 12207 Standard

The first group of documents from the SPICE project did not present a good compatibility with the ISO/IEC 12207 Standard (EMAM *et al.*, 1998). In later versions, care was taken to establish a better fit between SPICE and the international standard for life cycle software process, basically not to have two ISO standards presenting different concepts and definitions (EMAM *et al.*, 1998). Even so, ISO/IEC TR 15504 and ISO/IEC 12207 have structural differences, basically motivated by the difference in scope between one and the other.

Based on the above, the following considerations were made to enable the combined use of the ISO/IEC 12207 Standard with the future ISO/IEC 15504 Standard:

(a) The software life cycle processes to be considered are those defined by the ISO/IEC 12207 Standard, with the inclusion of the ISO/IEC TR 15504 Processes.

To attend to consideration (a), different ISO/IEC TR 15504 processes were analyzed, classified into five types: (i) Basic: processes identical to ISO/IEC 12207; (ii) Extended: expansions of ISO/IEC 12207 processes; (iii) New: processes outside of the scope of ISO/IEC 12207; (iv) Components: groups of one or more activities of the same ISO/IEC 12207 process; (v) Extended Components: groups of one or more activities of the same ISO/IEC 12207 process, with some additional concepts.

In this way, the “basic processes” and “extensions” of ISO/IEC TR 15504 were correlated with the ISO/IEC 12207 processes. The “component processes” were created to manage a greater level of detail in the evaluation of processes. This detail is not required in the scope of the definition of software processes and “component processes” and “extended components” are related with ISO/IEC 12207 activities or tasks. The “new processes” were added to the processes already defined by ISO/IEC 12207.

*(b) The base practices, scheduled in the evaluation model, are defined as activities and related ISO/IEC 12207 activities and tasks. These practices are related to the software processes in level 1, **Performed**, of ISO/IEC TR 15504.*

(c) The management practices are incorporated as activities to each software process from capacity level 1. These practices are also related to ISO/IEC 12207 activities and tasks.

2.2 Mapping CMM with the ISO/IEC 12207 Standard

Concentrating as it does on aspects related to software development CMM does not present the process structures defined by the ISO/IEC 12207 Standard, nor does it cover all the aspects related to the life cycle of a software product. The ISO/IEC 12207 Standard and CMM, however, have a similarity in their definition of the activity groups which should be present in a software process.

A key process area (KPA) defined by CMM, can contain activities present in more than one ISO/IEC 12207 process. However, it is possible to establish a relationship between these KPAs and ISO/IEC 12207 processes relating better to them (SEI, 1998; FERGUNSON and SHEARD, 1998).

Some CMM key process areas do not present a correlation, nor do they represent an extension of the processes defined by the ISO/IEC 12207 Standard (e.g. Defect Prevention and Requirements Management). With the objective of standardizing the proposed approach, fitting the CMM structure to the ISO/IEC 12207 standard, new life cycle processes were set up, related to the KPAs without correspondence with processes defined in ISO/IEC 12207. The new processes were given the same names as the original KPAs (table 1).

To allow the CMMs to be used combined with the ISO/IEC 12207 Standard, a mapping was done between the activities and the key practices scheduled in the key process areas, and the activities and tasks defined by the ISO/IEC 12207 Standard.

2.3 Life Cycle Processes and Mapping between the ISO/IEC 12207 Standard and the Maturity Models

This resulted in the proposal of new life cycle structure processes individually or collectively defined by ISO/IEC 12207, by the future ISO/IEC 15504 or by deriving from any key process area (KPA) of the CMM. Such processes have been divided into three great process classes, according to the ISO/IEC 12207 process classification. Table 1 illustrates the software life cycle processes.

So that the process definitions may adhere to the ISO/IEC 12207 Standard and the maturity models, a mapping was established among the processes, the activities and tasks contained in the Standard and in the models. Although they have distinct objectives, the ISO/IEC 12207 Standard as well as SPICE and CMM describe activities which should be incorporated in a software process. Referred models describe these activities using different definitions and detail levels, hindering their combined use in the process definition. Referred mapping is described in detail in MACHADO (2000). Part of the mapping which refers to the Quality Assurance Process is represented in Table 2.

Primary Processes	Supporting Processes	Organizational Processes
• Acquisition (ISC) • Supply (IS) • Requirements elicitation (SC) • Development (ISC) • Maintenance (IS) • Operation (IS)	• Documentation (ISC) • Configuration management (ISC) • Quality assurance (ISC) • Verification (ISC) • Validation (ISC) • Joint review (ISC) • Audit (ISC) • Problem resolution (ISC) • Defects prevention (C)	• Management (ISC) • Infrastructure (ISC) • Improvement (ISC) • Training (IC) • Project management (S) • Quality management (S) • Risk management (S) • Organizational alignment (S) • Human resource management (S) • Measurement (S) • Reuse (S) • Process change management (C) • Technology change management (C) • Quantitative process management (C) • Intergroup coordination (C)
Observations: (I) Process defined by ISO/IEC 12207; (S) Process defined by SPICE (ISO/IEC TR 15504) (C) Process derived from KPA of CMM.		

Table 1 – Software Life Cycle Processes

ISO/IEC 12207 Activities	ISO/IEC TR 15504 Base Practices (BP)	CMM 1.1 KPAs and Activities
• Process implementation (6.3.1).	• Develop quality assurance strategy (SUP.3.BP1). • Establish quality standards (SUP.3.BP2). • Define quality records (SUP.3.BP3). • Report quality results (SUP.3.BP6). • Handle deviations (SUP.3.BP7).	• KPA Software Quality Assurance - Activity 1. • KPA Software Quality Assurance - Activity 2. • KPA Software Quality Assurance – Activity 3. • KPA Software Quality Assurance – Activity 6. • KPA Software Quality Assurance – Activity 7.
• Product assurance (6.3.2).	• Assure quality of work products (SUP.3.BP5).	• KPA Software Quality Assurance – Activity 5.
• Process assurance (6.3.3).	• Assure quality of process activities (SUP.3.BP4).	• KPA Software Quality Assurance – Activity 4.
Observations: - The codification adopted is the one used by the standard and the models. - BP – Base Practice: a software engineering or management activity that, when consistently performed, contributes to achieve the purpose of a particular process (ISO/IEC TR 15504, 1998). - KPA – Key Process Area: Each KPA identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability (PAULK *et al.*, 1997). - SUP – Customer-Supplier Process Category (ISO/IEC TR 15504, 1998).		

Table 2 – Quality Assurance Process – Example (not detailed) of mapping among ISO/IEC 12207, ISO/IEC TR 15504 (SPICE) and CMM

3. Description of the Model for Definition of the Software Processes

The approach presented in section 2, for a combined use of the ISO/IEC 12207 Standard with the maturity models was incorporated in a generic model for definition of software processes for projects. The purpose of the model is to define and organize some of the main factors influencing the definition of a software process, structured so as to obtain automated assistance for process definitions.

When defining a software process, besides the ISO/IEC 12207 Standard and the maturity models (CMM or ISO/IEC TR 15504), the following elements are also considered: the characteristics of the organization (or only a part of the organization) for which the process is being defined, the type of software development environment for which the process is being defined (for example, DOSDE - Domain Oriented Software Development Environment) (OLIVEIRA *et al.*, 1999), the type of software (for example, information systems), the development environment (for example, object oriented), the development characteristics (for example, development with geographically distributed teams), the characteristics of the project and of team, and the quality characteristics of the product. The model organizes the factors involved in the definition of a software process in three steps: ***(i) Definition of the standard process, (ii) Specialization of the standard process, and (iii) Instantiation for projects.***

These steps result as software process products in different levels of abstraction. Figure 1 makes a schematic representation of the proposed model, where the steps involved in the definition of a software process to a project are presented from a standard process and the intermediate products obtained; as well as allocating the factors which influence the definition of the process at the levels of abstraction proposed by the model.

The model can also be used just for the definition of a specific process for a project, without considering the definition of a standard process. In this approach the same factors influencing the definition of a software process are considered, although they are not established intermediate products. Below are described the three steps involved in the definition of a software process for a project, considering the definition of a standard process.

3.1 Definition of the Standard Process

In an organization, diverse projects can coexist, each with specific characteristics. However, there is a group of fundamental elements which need to be incorporated in any defined process. EMAM *et al.* (1998) define this group of fundamental elements as the standard process, or in other words, the basic process which guides the establishment of a common process in the organization. In this way, a standard process defines a single structure to be followed by all the teams involved in a software project (MAIDANTCHIK *et al.*, 1999) independently from the software characteristics to be developed.

In the current literature, one notes a tendency to use standard process in the processes definition. The ISO/IEC 12207 Standard (1995) and the maturity models, SPICE (ISO/IEC TR 15504, 1998) and CMM (PAULK *et al.*, 1997), describe the definition of a standard process for the organization as an essential factor in the search for maturity in software processes, with the added requirement to obtain capability at level 3 for CMM as well as for ISO/IEC TR 15504. These standards and models also describe procedures to be used in the specialization of the standard process for a project.

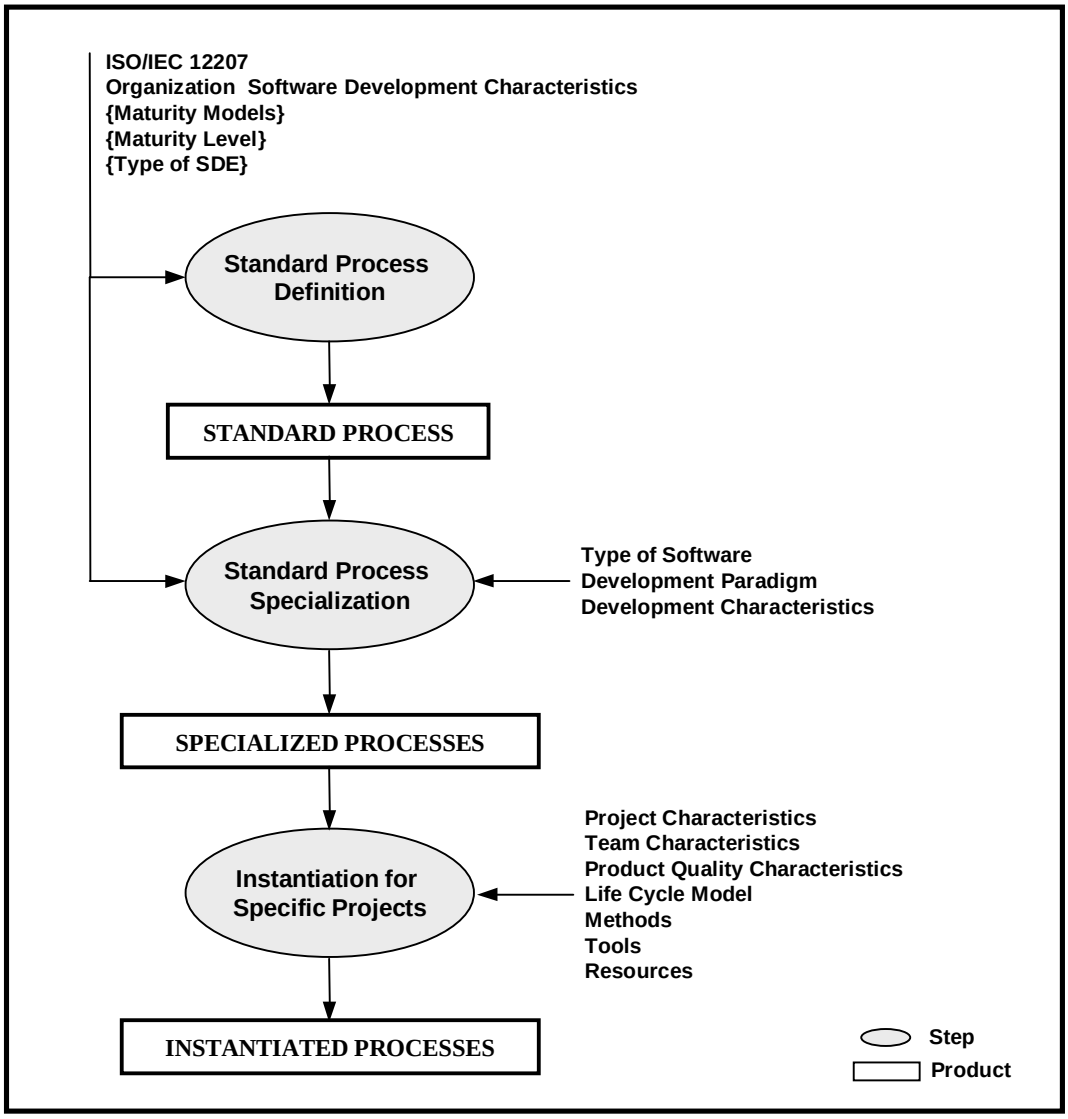


Figure 1 – Model for Definition of Software Processes

The ISO/IEC 12207 Standard and the Maturity Models

The definition of the standard process of the organization must use the ISO/IEC 12207 Standard and the software development characteristics in the organization. This definition may also consider one of the maturity models associated with the capability level of the organization in Software Engineering, and the type of SDE to be installed (SDE oriented or not to the application domain).

The standard process of the organization is constituted by a group of software processes (Table 1) relevant in the context of the organization projects. In this way, if an organization produces software for internal use, it becomes unnecessary to define a standard supply process.

For each software life cycle process incorporated in the standard process, the following should be defined (ISO/IEC TR 15504, 1998; OLIVEIRA *et al.*, 1999): the main objectives and criteria for concluding the process; the activities and sub-activities, with the identification of the required professional type (the professional types considered are: software engineers, designers, customers, project managers, operators and programmers); and finally, the products generated / consumed and resources required.

Software Development Characteristics in the Organization

The definition of the standard process also considers the software development characteristics in the organization. Considering that the main objectives of the software producer organizations are to define, use and establish continued improvements in their software processes, in order to reach these objectives it becomes fundamental that these organizations direct their forces in the application of the methods and practices of software engineering, and also take into consideration characteristics related to the work environment, to the knowledge and experience of the teams involved, and to their own culture and organization experience in developing software. When defining a software process, it is desirable that this is as appropriate as possible for the teams using it; in this way, it becomes important to incorporate the characteristics particular to each organization into the defined processes (ROCHA *et al.*, 1999).

Type of SDE

Finally, the standard process to be defined can be oriented to the domain, when defining the process for Domain Oriented Software Development Environments (DOSDE). With this, a specific activity, called ***Domain Investigation***, is incorporated into the standard process as a sub-activity of all the activities making use of the domain theory defined in the DOSDE (as for example, the analysis and project activities), with the objective to carry out the research and use the required domain in the corresponding activity (OLIVEIRA *et al.*, 1999).

3.2 Specialization of the Standard Process

The standard process is generic and shall be adapted to each project. This adaptation involves a great number of factors, including the choice of the development paradigm, the analysis of the characteristics of the project and of the team, the choice of the life cycle model, the methods and the tools. Then, the adaptation of the standard process to a project, as suggested by ISO/IEC 12207 and the maturity models, leads to a great variation of the abstract level, which might not be an interesting practice.

Therefore, the specialization step proposed as the second of the three-step approach consists of adopting the standard process to attend to different approaches of development, where the following factors are generally considered: (i) the type of software scheduled to be developed (for example, specialist systems or information systems); (ii) the development paradigm to be adopted (for example, object oriented or structured) and (iii) the development characteristics. Therefore, it is expected to reduce the abstract gap between the standard process and the project process.

When specializing a standard process, activities are added or modified, in accordance with the context of the purpose of the specialization. At the end of the specialization step, a software process is obtained adhering to the type of software and to the development paradigm to be adopted. The specialization considering a maturity model and a type of SDE can also be carried out, as long as these characteristics have not been incorporated during the definition of the standard process. In accordance with the model, the specialized process also allows its use by diverse projects, provided it is seen to incorporate characteristics which attribute to it a certain level of generality.

Type of Software

The type of software to be developed has an impact in the software process to be defined. An interactive software, for example, requires inclusion in the development process of activities

that optimize the man-machine interaction. With the ample use of the World Wide Web, and the expansion of e-business, security is a priority issue in interactive systems. In another approach, in the development of information systems, it becomes a fundamental issue to guarantee the adequacy of the business rules by the implemented procedures, as well as accuracy in the information records and issues of reliability and recovery after faults. Table 3 illustrates quality guarantee activities particular to two specific types of software (MCMANUS, 1999).

Interactive Software
• **Request Analysis Activity** ➢ Optimize the human-machine interaction, including commands and functions which improve response times ➢ Optimize interface and interaction aspects, including data types, text size, use of graphics and colours • **Project Activity** ➢ Promote inspections of project components to guarantee the adequacy of the interactive system for issues relating to receptivity and presentation on screen • **Verification Activity** ➢ Monitor the human-machine interactions, the use of menus, the "receptivity" on the part of users, interfaces, etc.
Information Systems
• **Request Analysis Activity** ➢ Establish requests for the case of interruption in the functioning of the system ➢ Establish requests for recovery from faults

Table 3 – Example of Activities to Assurance Quality in accordance with the Type of Software (MCMANUS, 1999)

Development Paradigm

In the specialization step, the development paradigm to be used should be chosen. This choice is strongly linked to the type of software to be developed and to the technology used. The choice of paradigm implies modifications in the definition of some activities of the standard process previously defined. The analysis activity undertaken under the structured paradigm uses a different approach to that used under the object oriented paradigm. Depending on the paradigm chosen, these adaptations are adopted during the specialization step of the standard process.

Development Characteristics (Specialization in accordance with the Model/Maturity Level)

The model and maturity level can be incorporated during the specialization step of the standard process, as long as this has been defined based only on the ISO/IEC 12207 Standard; in which case the maturity models are not used. This approach is particularly useful when defining software processes for projects where work teams have different software engineering skill levels. In this case, the standard process would define a single structure to be followed by all the teams involved in a software project (MAIDANTCHIK *et al.*, 1999), later specialized in accordance with the capability level of the teams involved.

3.3 Instantiation for Projects

Finally, the last step of our approach, named instantiation, consists of incorporating the specialized process characteristics related to the project, for which the process is being defined. The following issues shall be considered in the instantiation step: characteristics of the project, characteristics of the team, life cycle model, characteristics of product quality, methods, tools and resources (human, software and hardware).

The Characteristics of the Project and of the Team, and Selection of the Life Cycle Model

In the definition of a software process for a project, it is necessary to find the selection appropriate to a life cycle model. There are innumerous life cycle models in the literature and an inadequate selection has a direct influence on the success of a project. When selecting a life cycle model, the literature suggests a group of criteria influencing this selection, and values were attributed in order to evaluate the adequacy of a life cycle model to a project (table 5). These criteria and their values were applied to currently proposed life cycle models, and supply guidelines on the adequacy of a life cycle model to the characteristics of a specific project. Table 5 shows some of the criteria encountered in the literature (ALEXANDER and DAVIS, 1991; PRESSMAN, 1997; PFLEEGER, 1998; FALBO *et al.*, 1999).

In the instantiation step, the activities and sub-activities of the development process should be adapted to the chosen life cycle model. At this time, new activities will be able to be added, as is the case of "risk analysis", scheduled in the Spiral life cycle.

Characteristics of Product Quality

The ISO/IEC 9126 Standard (1991) defines six quality characteristics which should be evaluated in a software product, although, these characteristics don't need to be present in every software product. The evaluation of level in which referred characteristics should be present in a software product is determined by the objective of the application, and also by the evaluation of the users. In general, guaranteeing the presence of a quality characteristic in a determined level causes an increase in the cost of the project.

Based on evaluating the quality characteristics described in the ISO/IEC 9126 Standard in relation to the product to be developed, BOEGH *et al.* (1993) propose that it is possible to suggest techniques for evaluation of these characteristics. These suggestions are based on the level of importance which the software engineer and by the end-product user attribute to each quality characteristic, which will have an influence in the exactness of the evaluation techniques to be applied.

Thus, five levels of importance were defined for the quality characteristics defined by the ISO/IEC 9126 Standard. These levels are presented as follows: Not relevant, Little relevance, Relevant, Very relevant and Necessary. In a project, the levels of importance for each quality characteristic are determined during the survey of the requirements step.

During the instantiation step of a process for a project, the software engineer should attribute for each quality characteristic defined in the ISO/IEC 9126 Standard, the required level for the referred characteristic for the application to be developed. In accordance with the level attributed, quality evaluation techniques are included in the process to be defined. Table 4 shows the correspondence between the levels of importance and the evaluation techniques used for the "Functionality" quality characteristic.

Levels of Importance	Evaluation Techniques
Not relevant	No technique
Little relevance	Tests of "black box" type
Relevant	Tests of "black box" type and document inspections
Very relevant	Tests of "black box" type, document inspections, and tests of "white box" type
Necessary	Tests of "black box" type, document inspections, tests of "white box" type and formal proof

Table 4 – Evaluation Techniques used in accordance with the Level of Importance of "Functionality" Quality Characteristic in the Product

	Life Cycle Models				
	Waterfall	**Incremental**	**Evolutionary**	**RAD**	**Prototype**
Personal Criteria					
• Experience of the users in the domain of the application	Specialist	Experienced or Specialist	Any	Experienced or Specialist	Any
• Facility of the users in expressing requirements	High	Average or high	Any	High	Any
• Experience of the development team in the domain of the application	Experienced or Specialist	Experienced or Specialist	Any	Experienced or Specialist	Any
• Experience of the development team in software engineering	Any	Any	Any	Specialist	Any
Criteria related to the Problem					
• Level of maturity of the application domain	Established	Average or Established	Any	Established	Any
• Complexity of the problem	Simple	Simple or Difficult	Any	Low	Any
• Frequency of change of requirements	Low	Low or Average	Any	None	Any
• Level of magnitude of the changes in the requirements	None	Low	Any	None	Any
• Level of modularity of the problem	Any	Average or High	Average or High	High	Average or High
Criteria related to the product					
• Size of the application	Small	Any	Any	Any	Any
• Level of complexity of the application	Low	Low or Average	Any	Low or Average	Any
• Level of importance of the interface requirements	Low	Low	Low	Low	Any
Criteria related to the resources					
• Availability of human resources	High	Any	Any	Adequate or High	Any
• Level of access to the users	High	Average or High	Average or High	High	High
Criteria related to development					
• Applicable to the need of delivery of intermediary products.	No	Yes	Yes	No	Yes
• Level of technical risks.	Low	Low or Moderate	Low or Moderate	None	Low or Moderate
• Adopted paradigm.	Structured or OO	Structured or OO	Structured or OO	Structured or OO	OO

Table 5 – Criteria applied to Life Cycle Models

Methods, Tools and Resources

During the instantiation step, methods, tools and resources (software, hardware and human) should be defined for the project. This choice should be adherent for the paradigm chosen and the type of software to be developed.

4. Def-Pro Tool

The purpose of the Def-Pro tool is to define a software process for an organization, based on the ISO/IEC 12207 Standard, on the maturity models, on the capability levels, on the software development characteristics of the organization, on the type of SDE desired for instantiation, on the type of software to be developed, on the development paradigm adopted, on the characteristics of the project and team, and finally on the quality characteristics of the product to be developed. The construction of Def-Pro involved the following steps: (i) the inclusion of specific classes to represent the concepts of standards, maturity models, capability levels, organizations, software life cycle processes and standard processes; (ii) the conception of a structure integrating the concepts involved in the ISO/IEC 12207 Standard and in the maturity models; and (iii) the implementation of the Def-Pro tool.

The final tool product is the software process defined for an organization. The referred process is composed of one or more software life cycle processes, as well as adherence to the ISO/IEC 12207 Standard or to some maturity model or level. Figure 2 presents a simplified model of the classes storing the knowledge about standards and maturity models, as well as the software process defined by an organization.

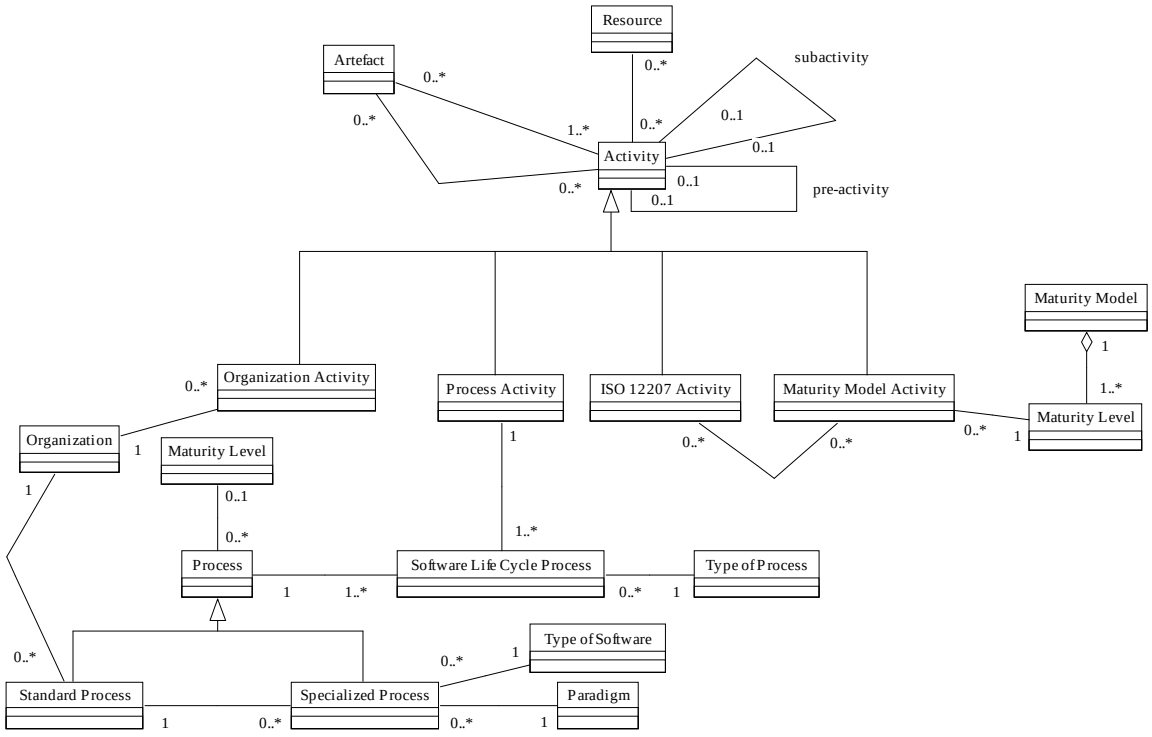


Figure 2 – Class Diagram

Figure 3 illustrates the definition step for a standard development process adherent to level 3 of the future ISO/IEC 15504 Standard. This tool requires the inclusion of activities relating to maturity level 3, and has an option to allow the software engineer to include level 4 and level 5 activities.

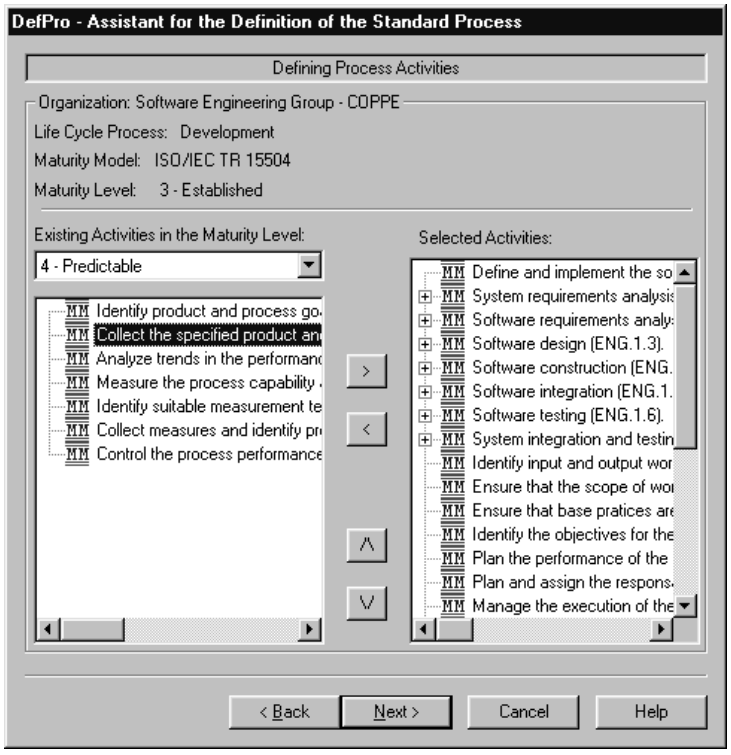


Figure 3 – Definition of a Standard Development Process adherent to Level 3 of the future ISO/IEC 15504 Standard

Based on the approach proposed in Section 2, the tool allows the software engineer to analyze each activity to be defined, comparing it with one or more activities equivalent to the ISO/IEC 12207 Standard (Figure 4). This has been requested by the organizations and was constructed based on the mapping between the ISO/IEC 12207 Standard and the maturity models. If desired, the software engineer can change some activities in the maturity model for activities in the Standard. This decision can be motivated by questions of nomenclature or by the need for a greater level of detail. The mapping also defines the activities pertinent to the ISO/IEC 12207 Standard and not contemplated in the maturity models.

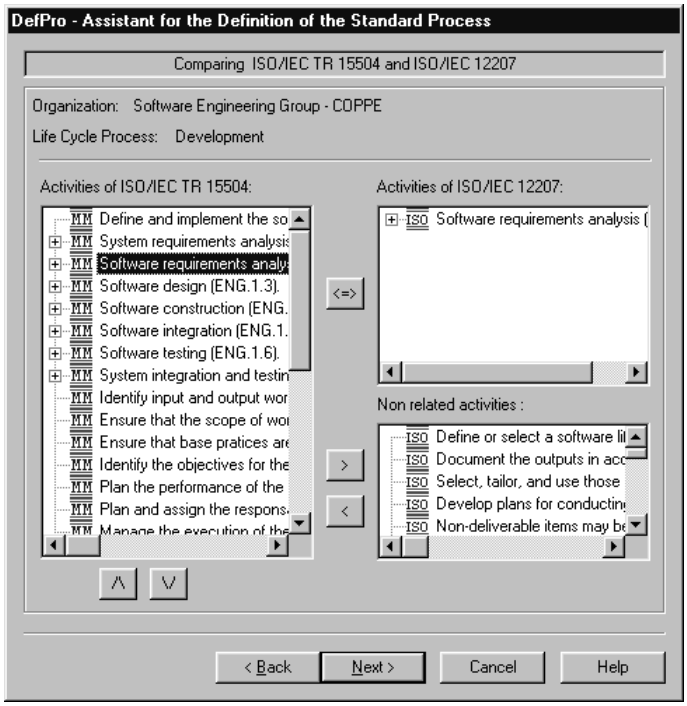


Figure 4 – Comparative analysis between the activities of ISO/IEC TR 15504 and ISO/IEC 12207

Finally, the defined development standard process activities, level 3 ISO/IEC TR 15504 are visualized (Figures 5 and 6). The coding presented corresponds to their use by the maturity model included in the definition.

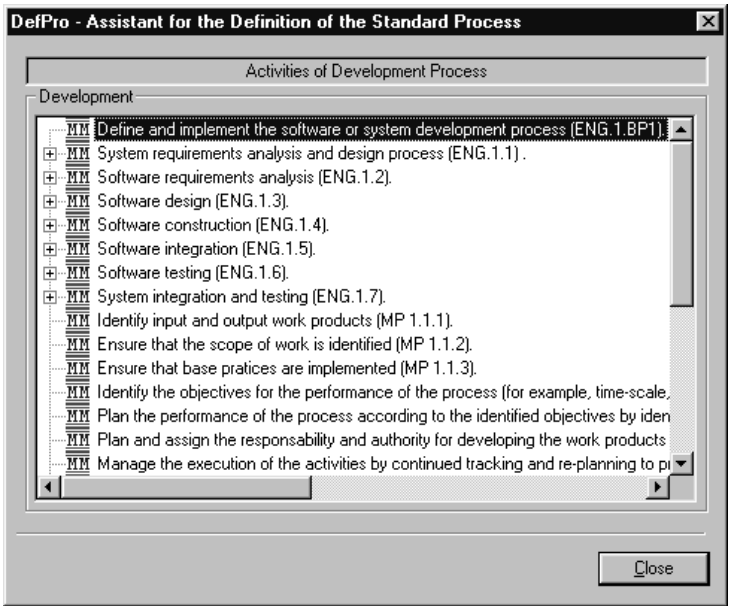


Figure 5 - Development Standard Process Activities

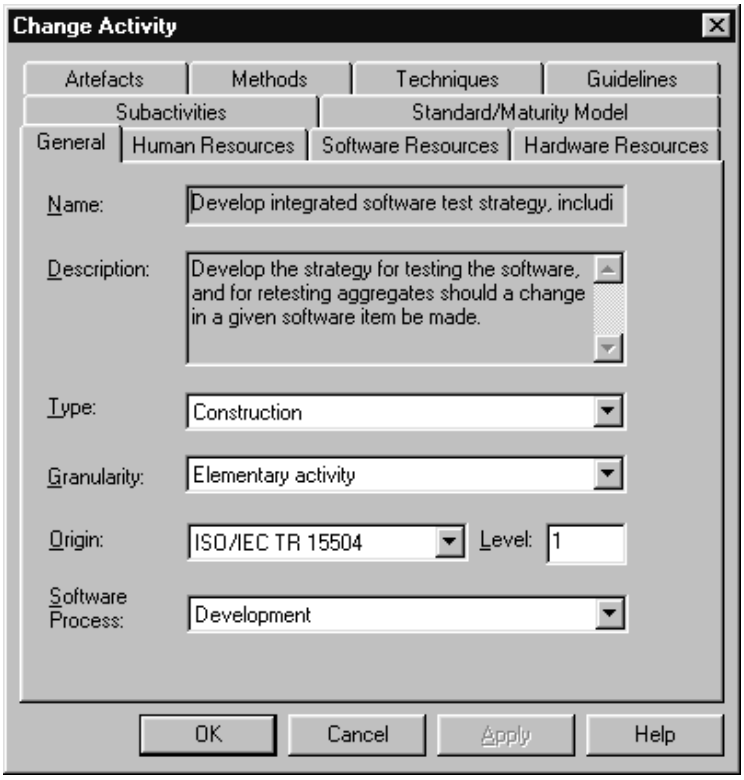


Figure 6 – Activity Definition

5. Conclusions

The definition and use of standards in software engineering are important practices for homogenizing software development in projects and organizations. With the advent of international standards and preoccupation with the quality of the software products being generated, the organizations adopted the ISO/IEC 12207 Standard and maturity models in the definition of software processes. In this sense, the ISO/IEC 12207 Standard as well as the maturity models establish the definition of a standard process as the condition for establishing a disciplined software process. The combined use of the Standard and the models, however, require an understanding of their differences and similarities, which implies difficulties in their use.

This paper presents an approach for the definition of software processes based on the need to define a standard process for an organization, and on the importance of adequacy of such process to the standards and maturity models. The model proposed in this paper considers the definition, specialization and instantiation of processes, allowing the definition of basic activities composing the standard process, as well as the definition of activities adequate under different contexts of development and software engineering capability levels.

To make this possible, a mapping of activities was developed between the ISO/IEC 12207 Standard and the maturity models (ISO/IEC TR 15504 and CMM), establishing a common vocabulary in order to enable the definition of software processes adherent to both standards. Referred mapping is an important subsidy for organizations using the approach of reaching adherence to the ISO/IEC 12207 Standard using the capability levels included in the maturity models.

To supply automated support to the definition of software processes based on the conceptual model proposed in section 3, a tool was implemented - Def-Pro. This tool uses the approach presented in section 2, supporting the combined use of the ISO/IEC 12207 Standard with maturity models, in the definition of software processes.

Other works in the context of software processes - whether or not using approaches based on knowledge - which take account of modeling, definition, analysis and simulation of software processes (CONRADI *et al.*, 1994; VALLETO and KAISER, 1996; FALBO *et al.*, 1999). The approach presented in this work extends these proposals, considering knowledge of standards and maturity models, as well as different levels of abstraction for software processes. The standard process is the first level of abstraction and represents a requirement for the organizations feeling the need to improve their software processes. There is as well as this the possibility to define other processes in the software life cycle, as well as the development process, an important characteristic which differentiates Def-Pro from the previous approaches.

The research in software processes points to the need to control and evaluate the processes during and after their execution. In this context, tools which assist measurement during the process are perspectives for the future. On the other hand, the evaluation of the level of extension executed by an activity is taken little advantage of and accompanies the tendency of the maturity models.

References

ALEXANDER, L. C., DAVIS, A. M., 1991, "Criteria for Selecting *Software* Process Models". In: *Proceedings of the Fifteenth Annual International Computer Software & Applications Conference – COMPSAC 91*, pp. 521-528, Tokyo, Japan, September.

BOEGH, J., HAUSEN, H. L., WELZEL, D., 1993, "A Practioners Guide to Evaluation of *Software*", In: *Software Engineering Standards Symposium*, Brighton, Inglaterra, Agosto.

CONRADI, R., HAGASETH, M., LARSEN, J-O., 1994, "EPOS: Object-Oriented and Cooperative Process Modelling". In: Finkelstein, A., Kramer, J., Nuseibeh, B. (eds), *Software Process Modelling Technology*, pp. 33-70, Advanced *Software* Development Series Research Press Ltd.

EMAM, K. E., DROUIN, J. N., MELO, W., 1998, *SPICE – The Theory and Practice of Software Process Improvement and Capability Determination*, IEEE Computer Society, Edwards Brothers Inc., Estados Unidos.

FALBO, R. A., ROCHA, A. R. C., MENEZES, C. S., 1999, "Assist-Pro: Um Assistente Baseado em Conhecimento para Apoiar a Definição de Processos de *Software*". In: *XIII Simpósio Brasileiro de Engenharia de Software*, Florianópolis, Santa Catarina, Brasil, October (in portuguese).

FERGUNSON, J., SHEARD, S., 1998, "Leveraging Your CMM Efforts for IEEE/EIA 12207", *IEEE Software*, pp.23-28, Setembro/Outubro.

ISO/IEC 9126, 1991, *Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their use*

ISO/IEC 12207, 1995, *Information Technology – Software Life Cycle Processes.*

ISO/IEC TR 15504, 1998, *Parts 1-9: Information Technology - Software Process Assessment.*

JACOBSON, I., BOOCH, G., RUMBAUGH, J., 1998, *The Unified Software Development Process*, Addison Wesley Longman, Inc.

MACHADO, C., DE OLIVEIRA, L., FERNANDES, R., 1999, "Experience Report – Restructure of Process based on ISO/IEC 12207 and SW-CMM in CELEPAR", In: *Proceedings of the 4 th IEEE International Software Engineering Standards Symposioum*, Curitiba, Paraná, Brasil, Maio.

MACHADO, L.F. D. C., 2000, *Modelo para Definição de Processos de Software na Estação Taba*, MSc Thesis, COPPE/UFRJ, March (in Portuguese).

MAIDANTCHIK, C., DA ROCHA, A. R. C., XEXEO, G. B., 1999, "*Software* Process Standardization for Distributed Working Groups". In: *Proceedings of the 4 th IEEE International Software Engineering Standards Symposioum*, Curitiba, Paraná, Brasil, Maio.

MCMANUS, J. I., 1999, "How does *Software* Quality Assurance Fit In ?". In: Shulmeyer, G. G., Mcmanus, J. I. (eds), *Handbook of Software Quality Assurance*, 3 ed., chapter 2, Prentice Hall PTR.

OLIVEIRA, K.M, ROCHA, A.R, TRAVASSOS, G. T., MENEZES, C., S., 1999, "Using Domain-Knowledge in *Software* Development Environments". In*: Software Engineering and Knowledge Engineering - SEKE 99*, Kaiserlautern, Alemanha, Junho.

PAULK, M. C., WEBER, C. V., CURTIS, B., CHRISSIS, M. B. (eds), 1997, *The Capability Maturity Model: Guidelines for Improving the Software Process*. Carnegie Mellon University, *Software* Engineering Institute, Addison-Wesley Longman Inc.

PFLEEGER, S. L., 1998, *Software Engineering Theory and Practice*, Prentice Hall PTR.

PRESSMAN, R. S., 1997, *Software Engineering: A Practitioner's Approach*, 4th edition, McGraw-Hill.

ROCHA, A. R. C., MAIDANTCHIK, C., OLIVEIRA, K. M., TRAVASSOS, G. H., XEXÉO, G., 1999, "Experience in Defining, Using and Improving *Software* Process". In: *Technical Report ES-507/99* - COPPE/UFRJ (in Portuguese).

ROUT, T. P., 1996, "The SPICE Project: Past, Present and Future", *Software Process '96*, Brighton, Dezembro.

ROUT, T.P., 1998, "SPICE and the CMM: is the CMM compatible with ISO/IEC 15504 ?", *AquIS'98*, Venice, Março.

SEI, 1998, "Top-Level Standards Map - ISO 12207, ISO 15504 (Jan 1998 TR), *Software* CMM v1.1 and v2 Draft C". URL: http://www.sei.cmu.edu

VALETTO, G., KAISER, G. E., 1996, "Enveloping Sophisticated Tools into Process-Centered Environments", Automated *Software* Engineering Notes, v. 3, n. 3/4, pp. 309--345, Agosto

ZAHRAN, S., 1998, *Software Process Improvement*, Addison Wesley Longman Inc.

QWE2000 Vendor Technical Presentation VT7

Mr. Matthew Brady
(McCabe & Associates UK Ltd)

"How to Test Better - Not Test More"

Key Points

- Most test programs are focussed on verifying that software conforms to the specification, and operates without error under expected conditions.
- However, functional combination means that there is almost an infinite number of possible tests for any given application, and that the vast majority of these tests will be of little value.
- Additionally, the assumption that testing based on functionality will be sufficient is incorrect û in many software projects a full functional test program will often cover less than 60% of the code.
- Other techniques must be utilised to improve the validity and completeness of functional tests, without adding a huge burden of new tests.

Presentation Abstract

Functional testing is often thought to be a straightforward logical process of converting functional specifications into test scenarios.

However there are several major challenges faced by functional testers.

The initial challenges are associated with the task of deriving a full functional test set and verifying that functional testing is complete. These challenges are in fact impossible to overcome because functional combination means that there are almost infinite scenarios which can applied during testing. It is therefore crucial to ensure that a minimum set of functional tests is executed providing maximum benefit.

Secondly, functional testing is very frequently under considerable time pressure due to slippage in earlier stages of the development process. This means that testing often has to be cut short in order to deliver software on time. It is therefore critical to ensure that tests are prioritised with important tests being performed early.

Additionally, the architecture and implementation of a software project influences the nature and scale of the required testing.

How can we improve our testing to ensure that the critical parts have been adequately tested without doubling or trebling the number of tests required?

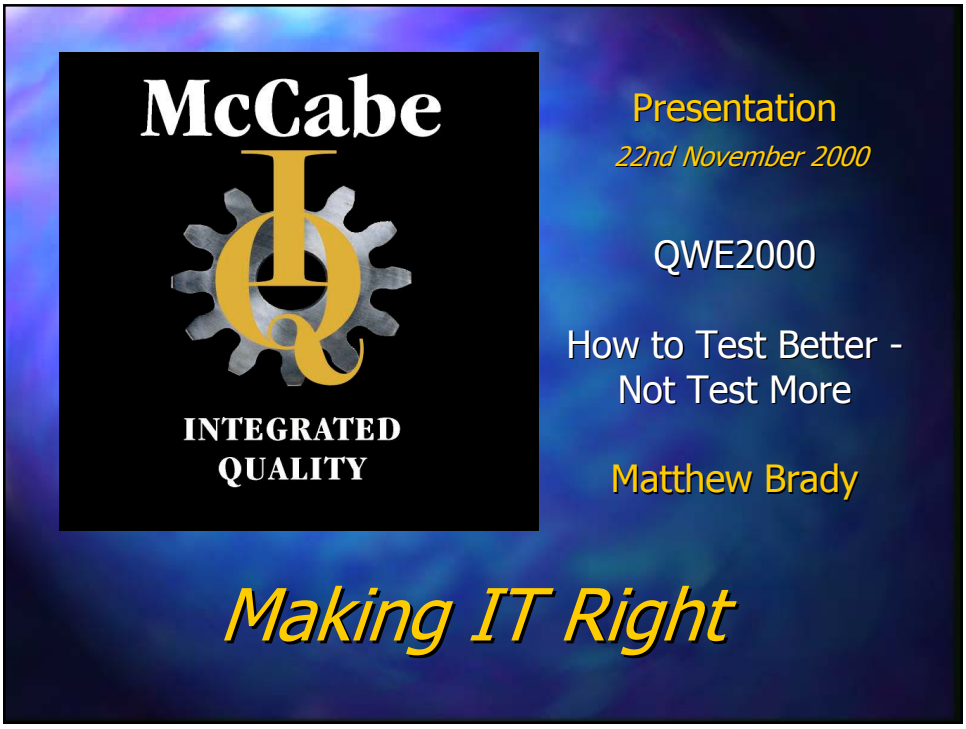
McCabe & Associates promote powerful software analysis techniques and tools to assess the risk associated with a particular test schedule and identify necessary improvements. These techniques will be outlined during the presentation.

About the Speaker

Matthew Brady has worked for McCabe & Associates in the UK distribution office for four years. During that time he has been involved in the implementation of McCabe solutions in many organisations across Europe including the UK National Air Traffic Centre, NEC, Nortel, Telefonica and Pace. He now heads up the consultancy and support team for McCabe in Europe.

Prior to McCabe, he worked for Kinesix Corp. delivering real-time graphics solutions to the aerospace and safety critical markets.

He graduated in 1987 in Mathematics from Manchester University

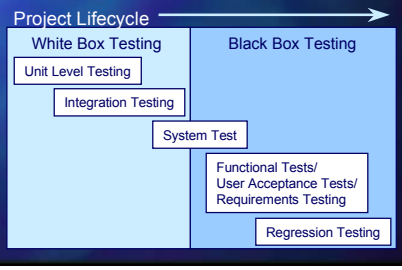


McCabe & Associates

- 20 Year History
- Software Analysis Tools:
  - Software Measurement
  - Testing Improvement
  - Comprehension
- Configuration Management Tools

Testing Software

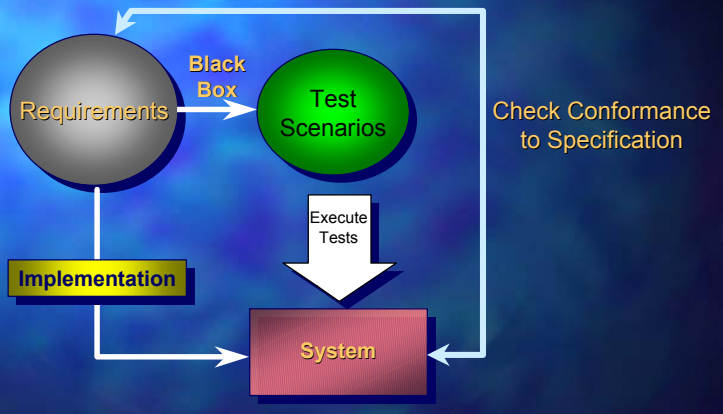
- How Can Software be Tested?
  - With Knowledge of Functionality
    - Black Box
  - With Knowledge of the Code
    - White Box
    - Developer Testing



McCabe & Associates

3

Software Testing
functional testing

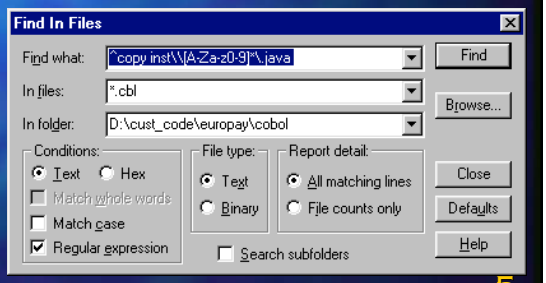


McCabe & Associates

4

Software Testing

- Deriving Functional Tests
  - How Many Combinations Exist?
  - Which Combinations are Important?
  - How Many Combinations are there in Your Applications?
  - What is Good Enough?

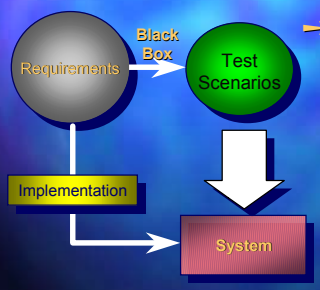


McCabe Associates

5

Functional Testing

- Derive Tests from Requirements:
  - → Not Always Possible for Complex Systems
  - → Functional Tests Are Often Highly Duplicate

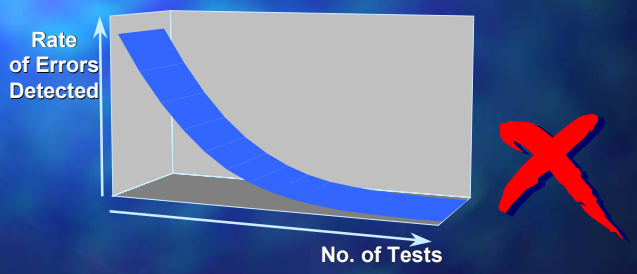


McCabe Associates

6

Software Testing

- When do you Stop Testing?
  - When Time Runs Out!
  - Testing Boredom



McCabe Associates

7

Regression Testing

- Change One Line of Code:
  - How Many Tests Should Be Performed?
  - Should Complete Application be Tested Again?
  - What is the Impact of a Change?

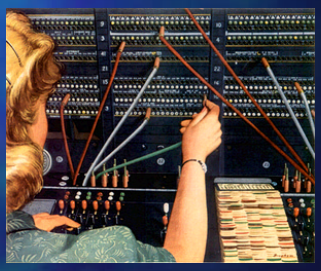
```
#include "fred.h"

void
main(char **argv, int argc)
{
    int i;

    for (i=0; i++; i<=100)
        printf("Value is %d\n", i);
}
```

McCabe Associates

8

Testing Software

- How Do People Test Functionally?
  - Manual Execution
  - Using "Capture / Replay" Tools
  - Automation / Scripting

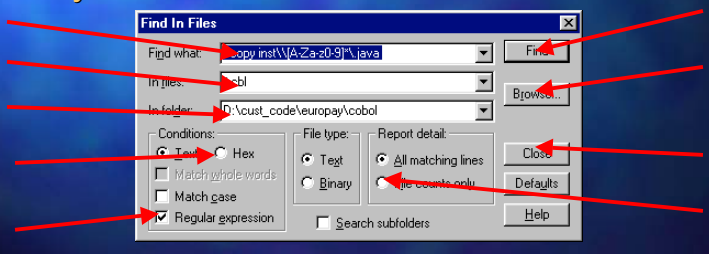


McCabe Associates

9

Testing Software

- What is Capture/Replay?
  - Run More Tests - More Quickly - More Easily
  - Ability to Check Outputs
    - Determine Test Outcome
  - Many Tools Available

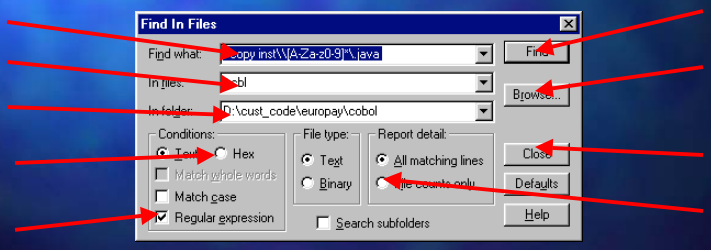


McCabe Associates

10

Software Testing

- Test Automation
  - Only Solves One Problem
    - Ability to Re-Run Tests / Check Output
  - No Indication "When to Stop"
  - Little Assistance with Deriving Tests



McCabe & Associates

11

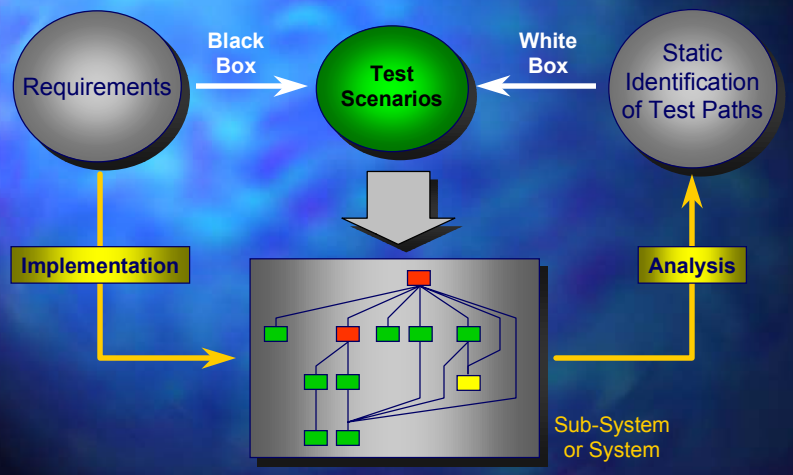
Common Testing Challenges

- Deriving Tests
  - Creating a "Good" Set of Tests
- Running Tests
- Verifying Tests
  - Verifying that Enough Testing was Performed
  - Providing Evidence that Testing was Good Enough
- When to Stop Testing
- Prioritising Tests
  - Ensuring that Critical or Modified Code is Tested First
- Reducing Test Duplication
  - Identifying Similar Tests That Add Little Value & Removing Them

McCabe & Associates

12

Software Testing
focussed testing



McCabe & Associates

13

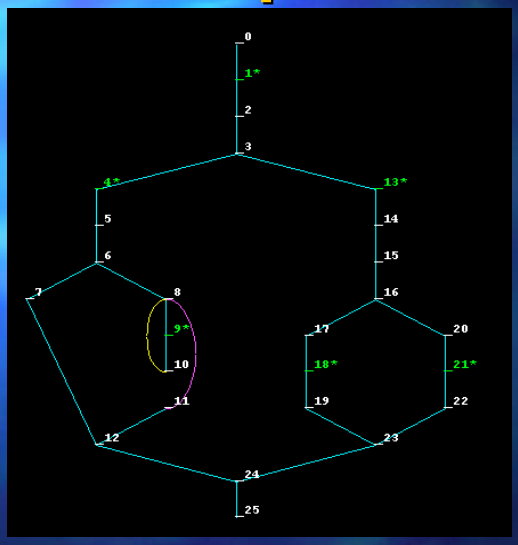
Software Testing
Deriving Unit Level Tests

```
    function_test(y)
0   {
        x=3;
1       if ( y < 4 )
2           x=sin(y);
        else
3           x=cos(y);
4       x=x*x;
5   }
```

McCabe & Associates

14

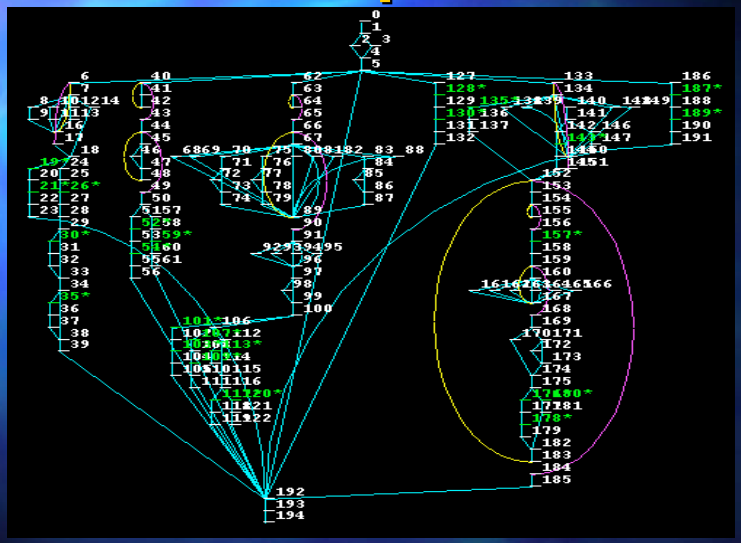
Example 1



McCabe Associates

15

Example 2

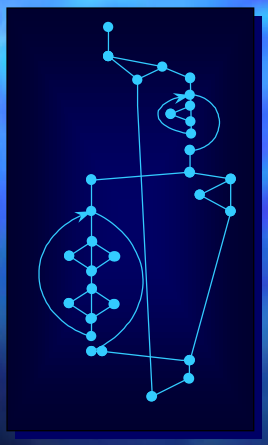


McCabe Associates

16

Software Testing

Test Derivation

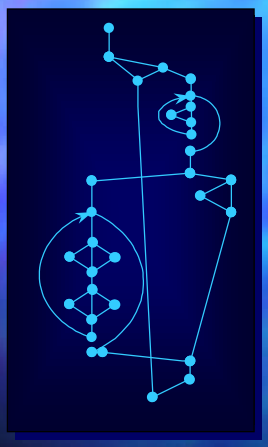


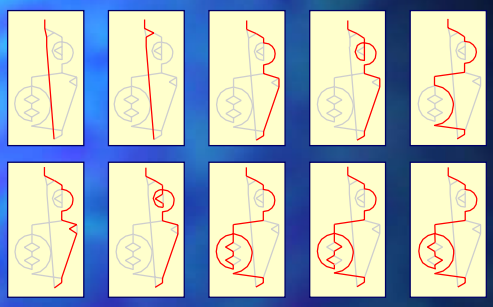
- Flowgraphs Visualise Logic
- Valuable for:
  - Comprehension
  - Test Derivation
- How Can Tests be Derived Using Flowgraphs?

McCabe & Associates

17

Software Testing





- Complexity = 10

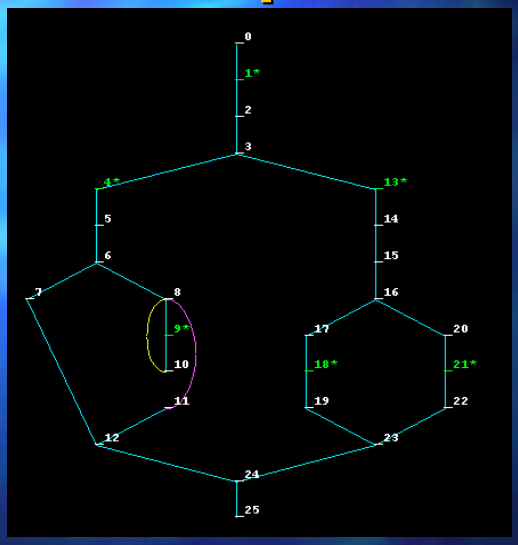
Means that 10 Minimum Tests will:

- Cover All the Code
- Test Decision Logic

McCabe & Associates

18

Example 1

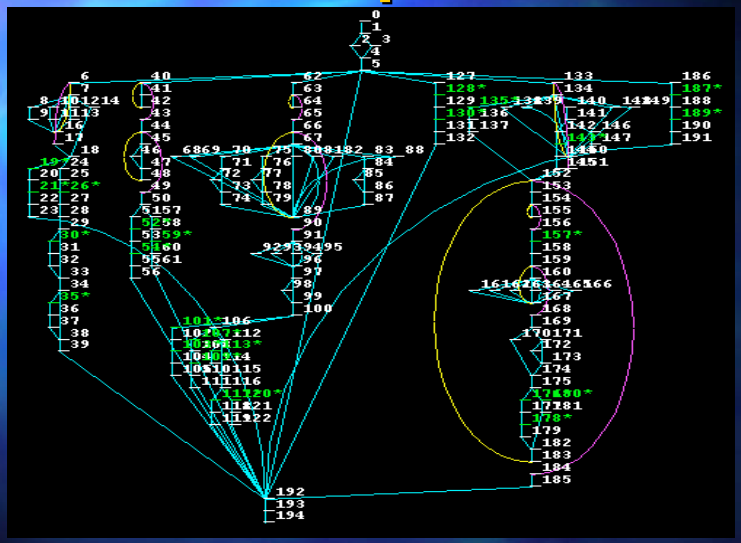


Complexity = 5

McCabe Associates

19

Example 2



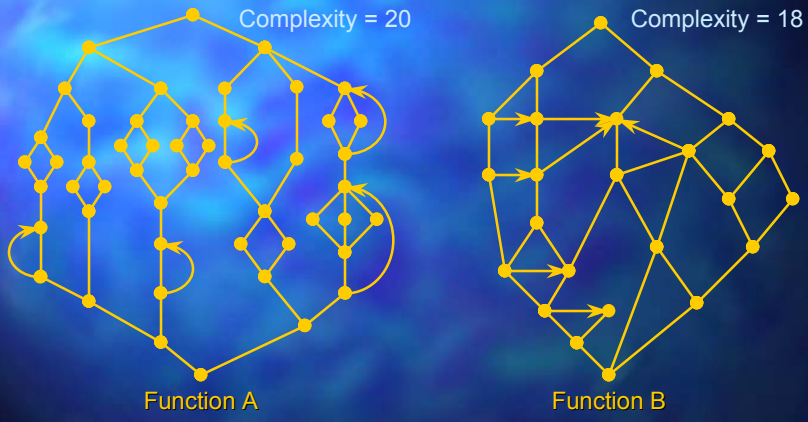
Complexity = 57

McCabe Associates

20

Software Testing

Complexity and Structure

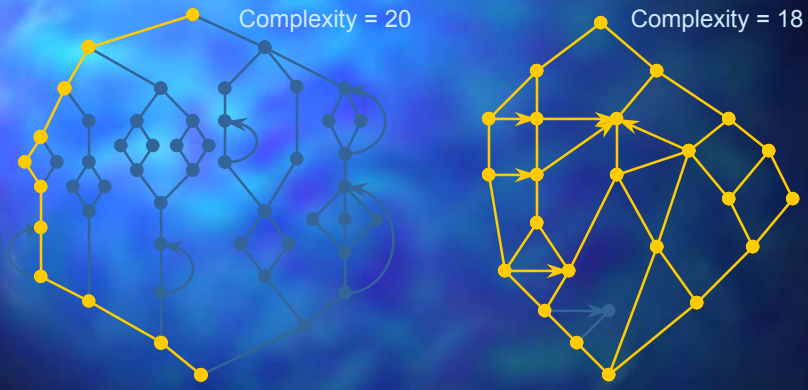


McCabe & Associates

21

Software Testing

Essential Complexity



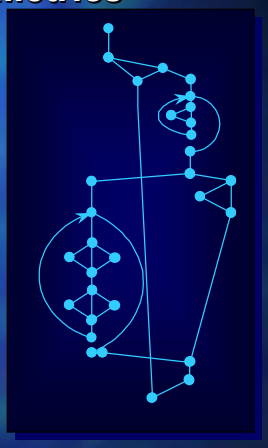
McCabe & Associates

22

Software Testing

McCabe Unit Level Metrics

- Complexity
  - Simple / Scalable
  - Language Independent
  - Easy to Measure
  - Indication of Testing Effort (No. of Tests)
- Essential Complexity
  - "Unstructure"
  - Indicates Testing Difficulty



McCabe & Associates

23

OO & Complexity?

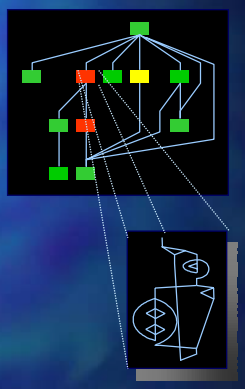
- Is Complexity Valid for OO?
  - Complexity Should be MUCH Lower in OO Systems
  - High Complexity Occurs When Object Model is Arcane / Poorly Implemented
- What About Essential Complexity?
  - Essential Complexity Should More Often be At a Minimum
  - High Essential Complexity Indicates Structural Programming

McCabe & Associates

24

Building from the Unit Up

- Execution Hierarchy
  - Entire Program
- Unit of Code
  - Function
  - Procedure
  - Subroutine
  - Method

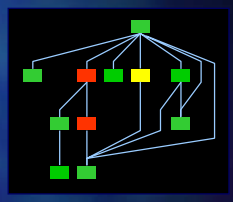


McCabe Associates

25

Building the Tree

- Procedural Languages
  - C / Fortran/ COBOL
  - Execution is Everything
  - Execution Tree is Simple to Determine
- Object Oriented Languages
  - C++ / Java / Ada95
  - What About Class Interaction / Overloading / Late-Binding?

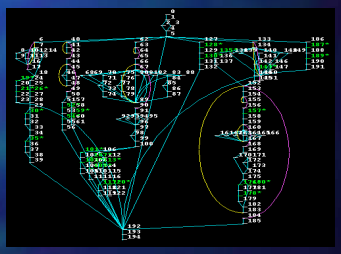


McCabe Associates

26

Test Improvement

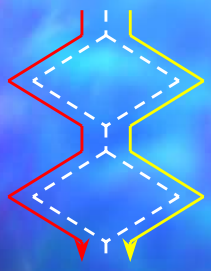
- Theoretical Testing Measures
  - Complexity
  - Essential Complexity
  - Program Complexity
  - Other Measures
- Real Testing Measures
  - Coverage
  - What is Coverage?



McCabe & Associates

27

Code Coverage



2 Tests Required

```
test_func(int a, int b)
{
    if (a==b)
        op1();
    else
        op2();

    if (a==c)
        op3();
    else
        op4();
}
```

Total number of executable lines of code = 11

Test Paths Listing

TEST 1	1 (10) :	if (a==b) => False
	6 (18):	if (a==c) => False
TEST 2	1 (10) :	if (a==b) => True
	6 (18):	if (a==c) => True

Test Coverage Results

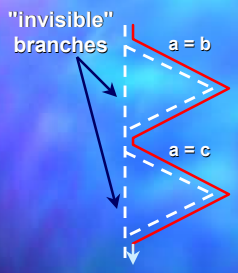
Test 1 : 7 Lines Tested => 64 %
Test 2 : 9 Lines Tested => 82 %

Test 1 & 2 : 11 Lines Tested => 100 %

McCabe & Associates

28

Code Coverage Limitations



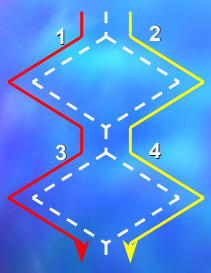

```
test_func(int a, int b)
{
    if (a==b)
        op1();

    if (a==c)
        op2();
}
```

- Positive Testing
  - Test Application to Execute Some Functionality
- Negative Testing
  - Ensure that Application Does NOT Perform Undesired Functionality
- 100% Code Coverage ONLY Tests the Execution of Lines of Code

McCabe & Associates

29

Branch Coverage



2 Tests Required

```
test_func(int a, int b)
{
    if (a==b)
        op1(); ->Branch 1
    else
        op2(); ->Branch 2

    if (a==c)
        op3(); ->Branch 3
    else
        op4(); ->Branch 4
}
```

Test Paths Listing

TEST 1	1 (10) :	if (a==b) => False
	6 (18):	if (a==c) => False
TEST 2	1 (10) :	if (a==b) => True
	6 (18):	if (a==c) => True

Test Coverage Results

Test 1 : 2 Branches Tested => 50 %
Test 2 : 2 Branches Tested => 50 %

Test 1 & 2 : 4 Lines Tested => 100 %

Total Number Branches = 4

McCabe & Associates

30

Coverage Comparison

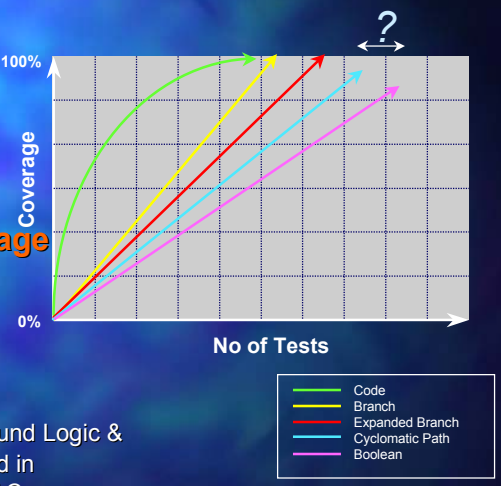
Code Coverage
Achieves High Results With Minimum Effort - Little Use

Branch Coverage
Relates More Linearly to Number of Tests, & Increases the Effectiveness of Testing

Expanded Branch Coverage
Includes Compound Logic

Path Coverage
Is More Rigorous, Requires Greater Effort & Cannot Be 100%

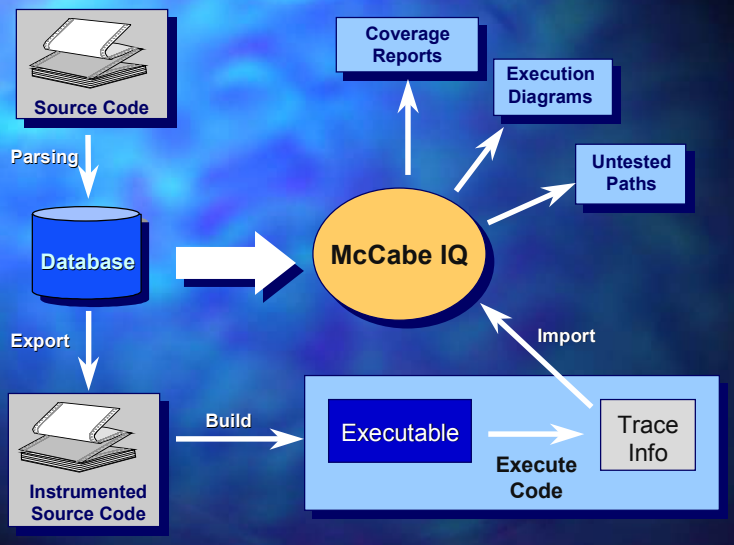
Boolean Coverage
Adds Rigorous Testing of Compound Logic & Cannot Be 100%. Should be Used in Conjunction with Another Form of Coverage



McCabe Associates

31

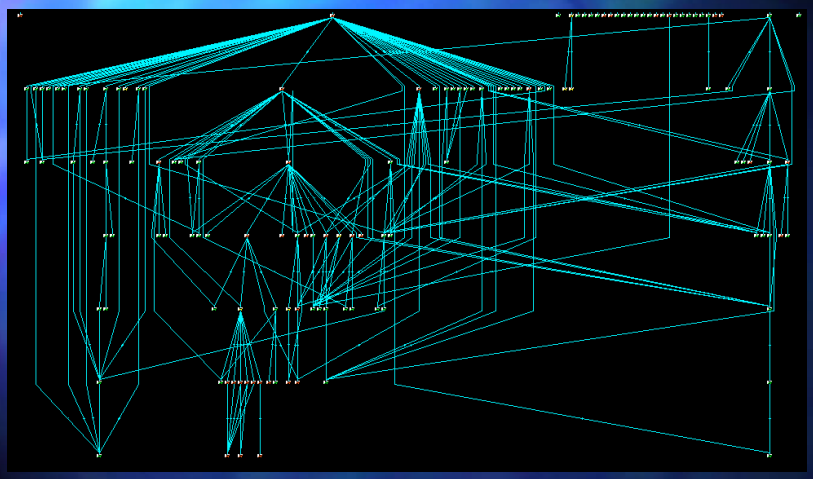
What is McCabe Test?



McCabe Associates

32

Focused Software Testing

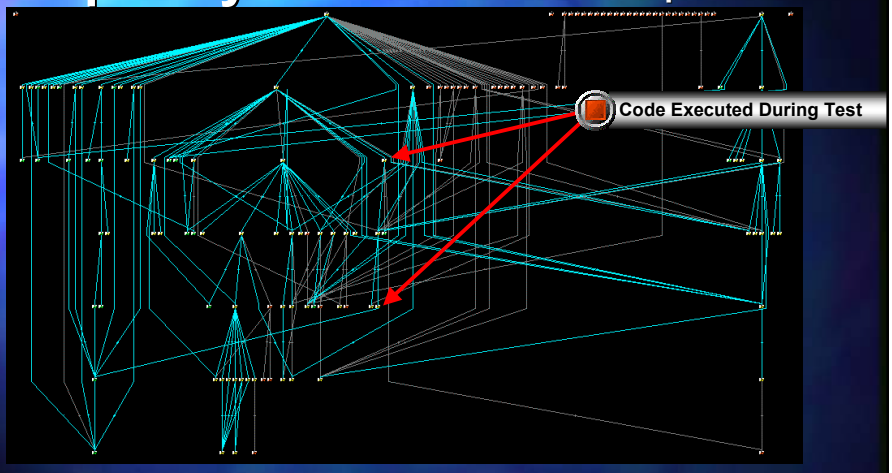


McCabe Associates

33

Focused Software Testing

quickly visualise execution paths

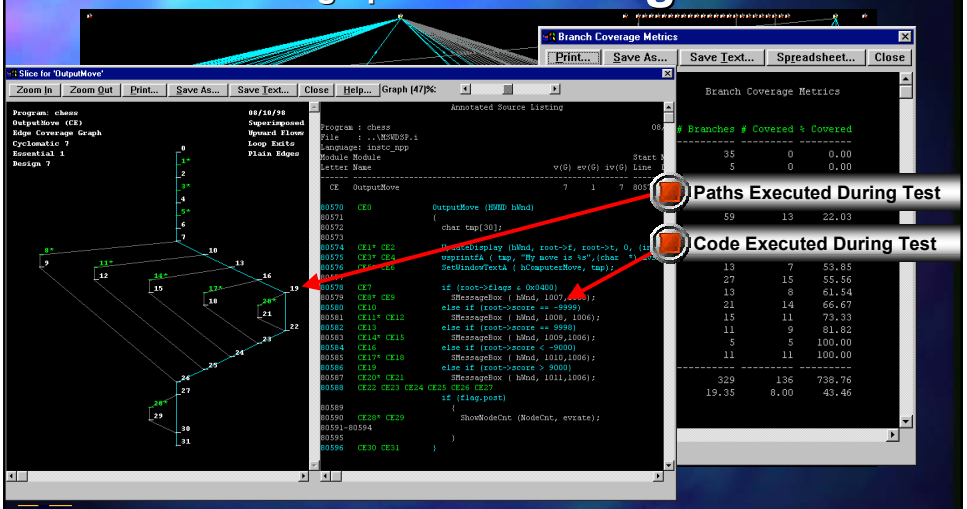


McCabe Associates

34

Focused Software Testing

full graphical coverage

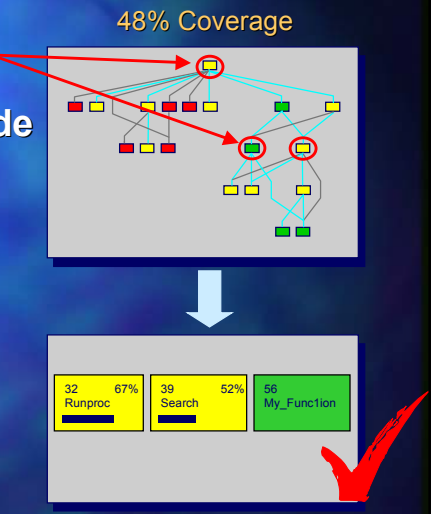


McCabe Associates

35

Using Coverage

- Determine "Critical" Code
  - Complex/Unstructured?
  - Recently Modified?
  - Functionally Important?
- Assess Coverage for "Critical" Code
  - Coverage Report for "Critical" Group
  - Examine Untested Branches



McCabe Associates

36

Conclusions

- Assess Testing Risk
  - Measure Software Components
- Derive Initial Tests Using Black Box Techniques
  - Functional Testing
- Examine Execution Achieved
  - Measure Coverage
- Derive Additional Combinations
  - Examine Untested Hierarchies / Units
- Focus on "Important" Code
  - Do Not Try to Cover All The Code

McCabe Associates

37

McCabe IQ

INTEGRATED QUALITY

Presentation

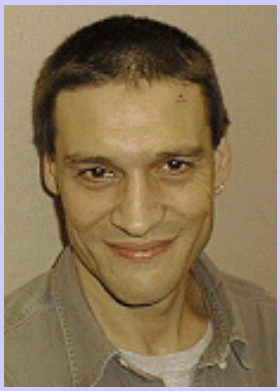
22nd November 2000

QWE2000

How to Test Better - Not Test More

Matthew Brady

Making IT Right



QWE2000 Session 8T

Tobias Mayer
(eValid, Inc.)

Browser-Based WebSite Testing Technology

Key Points

- 99% of Websites (html, etc. pages) are viewed from a Browser.
- In testing such Websites, the Browser-based testing method is recommended for accuracy and reliability.
- The talk will demonstrate: Load Testing, User Interactivity Testing, Content Validation & Verification, Download Timing, Rendering, and Security/e-commerce Issues in this context.

Presentation Abstract

Because 99%+ of Websites (html, etc. pages) are viewed from a Browser the 'Browser-Based Website Testing' approach is recommended as the best way of testing a Website. This approach allows the Tester to view (and examine) a website exactly as a User will view it. The User is the essential ingredient in the whole Website/Viewer algorithm. Without the user, the Website has no useful purpose - Full Stop! Other approaches offer only partial solutions to the Website Testing problem. The Browser-based approach offers a full solution. This talk will promote 'Browser-based' technology and offer demonstrations of its practical application.

About the Speaker

Tobias Mayer is a senior software engineer at Software Research, Inc. He is reponsible for the main design and implementation of the "eValid" Web Test engine. Tobias has a (UK) BSc from South Bank University, London. He is a member of, and OO Metrics consultant to, the Center for Systems & Software Engineering (CSSE) at South Bank University. Tobias has presented and published a number of papers on OO metrics, including papers at IEEE 'TOOLS' 1999 and British Computer Society 'SQM' 1999. During this year, Tobias has presented a number of seminars on Website Testing strategies in the UK. He also presented the "Quickstart - Website Testing" seminar at the 'Quality Week 2000' conference in San Francisco, June 2000.

Tobias Mayer
eValid, Inc.
email: mayer@soft.com

Browser-Based Web Site Testing Technology

Summary of presentation

1. Testing a site prior to uploading
   (~5 mins)
2. Testing a site after uploading
   (~10 mins)
3. Testing Strategies …
   (~15 mins)
4. A Test-Enabled Web Browser
   (~15 mins)

Running time = 45 minutes

Stage 1: Before the upload

- What can be tested?
  - » HTML
  - » Links
  - » Applets
  - » Scripts
  - » Images
- Other Considerations...

Stage 1: Before the upload

- What can be tested?
  - **HTML**
    - » Does it meet company/self-imposed formatting?
    - » Use of WebLint, HTMLTidy, etc.
    - » Spell Checkers
    - » Can HTML be tested like regular source code?

Stage 1: Before the upload

- What can be tested?
  - **Scripts**
    - » Again, do they meet company/self-imposed formatting?
    - » Syntax...
    - » Logic...

Stage 1: Before the upload

- What can be tested?
  - **Links**
    - » do they go where they are supposed to go?
    - » Can we actually test this prior to upload?
      - Own site -v- some other site...
    - » What kind of error messages can we get?

Stage 1: Before the upload

- What can be tested?
  - **Applets** (and other embedded executable objects)
    - » To what extent does an object interact with the HTML?
    - » Do objects execute correctly -
      - As independent entities
      - In relation to the Web site
    - » Executable objects should be fully tested prior to their inclusion on the site.

Stage 1: Before the upload

- What can be tested?
  - **Images**
    - » Do all images (gifs, jpegs, etc) exist at the correct locations?
    - » Does each image have the Width & Height values set?
    - » Check image sizes (i.e. byte size)
      - Is an image going to seriously affect the download time?
      - What methods can be used to reduce an image's size?

Stage 1: Before the upload

- Other Considerations…
  - Where will the site be positioned?
    - » Internet
    - » Intranet
    - » Local, browser-based app?
  - Other considerations …?

Stage 2: After the upload

- User Interactions
- Static / Dynamic Pages
- Test the site on a regular basis
- Use Automated testing

Stage 2: After the upload

- User Interactions
  - What will the User try to do?
  - Can the developer anticipate ***everything?***
    - » Every possible user interaction?
    - » Are users always smart???
  - How can the developer deal with "user stupidity?"

Stage 2: After the upload

- Static / Dynamic Pages
  - What is the difference?
  - Which is harder to test. Why?
  - Can dynamic pages actually be tested?
    - » How. What can be validated?
    - » How often?
    - » What can we learn?

Stage 2: After the upload

- Test the site on a regular basis
  - Transaction Testing
  - Content Validation
  - Link Checking
  - Load Testing
  - ...

Stage 2: After the upload

- Use Automated testing
  - Manual Testing:
    - » How many testers will be employed?
    - » How useful are they?
    - » What can they do that a robot-tester cannot do?

Stage 3: Testing Strategies

- Server-Side Testing
- Client-Side Testing
  - Browser-Level Testing
  - Operating System-Level Testing
- What else is there…?

Stage 3: Testing Strategies

- Server-Side Testing
  - Load Testing
    - » How many hits per (sec/millisec?)
    - » How many KB delivered per (sec/millisec?)
  - Site Validation
    - » Can it be done at this level?

Stage 3: Testing Strategies

- Client-Side Testing
  - How many times can you hit the server?
    - » What does this tell you?
  - Validations:
    - Visible Text
    - Table Cells
    - Images,
    - etc...

Stage 3: Testing Strategies

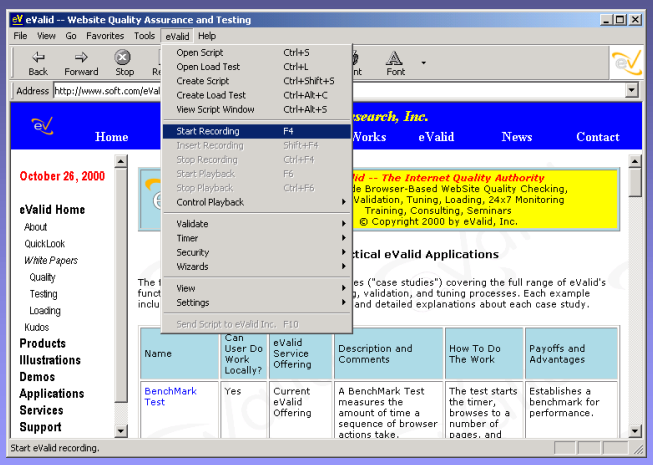
- Client-Side Testing
  - Passive Sites
    - » Provide information
    - » One-way 'communication'
  - Interactive Sites
    - » Provide & require information
    - » Two-way communication

Stage 3: Testing Strategies

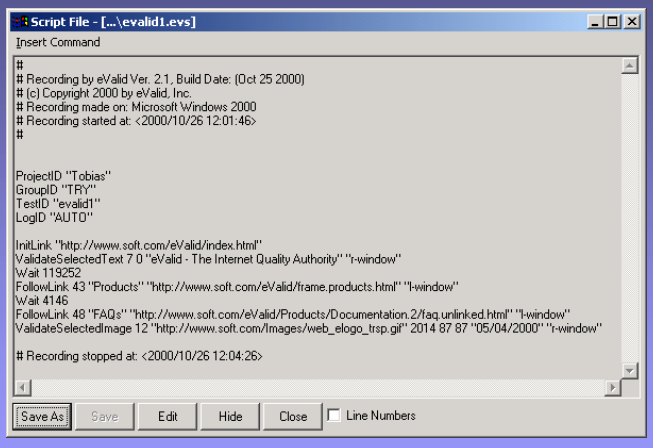
- Client-Side Testing
  - Static Sites
    - » Nothing changes from day to day unless the developer explicitly changes it.
  - Dynamic Sites
    - » Stuff changes all the time. Once created the developer has little, or no, control...

Stage 4: A Test-Enabled Web Browser

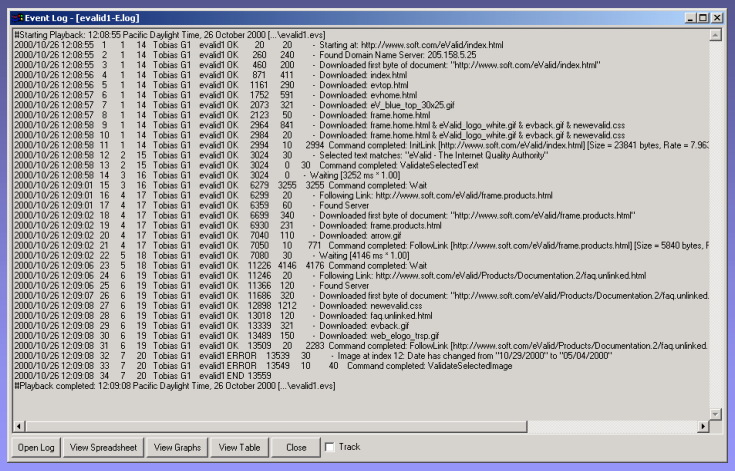
Stage 4: A Test-Enabled Web Browser



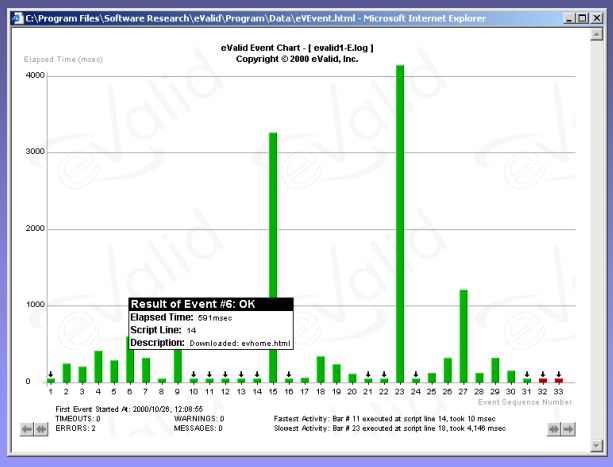
Stage 4: A Test-Enabled Web Browser



Stage 4: A Test-Enabled Web Browser



Stage 4: A Test-Enabled Web Browser



Conclusions

1. Web sites become more and more complex … hourly!
2. The challenge to test these sites becomes greater and greater.
3. Client-side testing is best done from the ***client perspective***

Therefore...

4. Client-side testing is best done from a Web Browser.

QWE2000 Session 8A

Mr. Gunthard Anderer [Germany]
(CMG ORGA - Team GmgH)

"Testing E-Commerce Systems - Requirements And Solutions (8A)"

Key Points

- The change of quality - requirements for e-commerce applications
- Discussion of the architecture of an complete e-commerce system
- The CMG TestFrame Environment as a solution
- Partners and their services
- Remaining problems

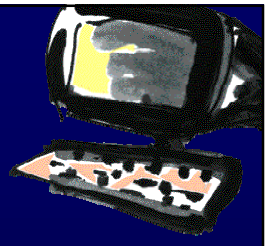
Presentation Abstract

E-Commerce systems require higher quality and permanent monitoring. In this paper the requirements are shown and an architecture for an automated testing - and monitoring environment for e-commerce systems (including internet -, legacy- and third party - applications) is proposed.

About the Speaker

Gunthard Anderer, Dipl.-Math., born 1951 in Nürnberg. Studied Mathematics and Physics at University Erlangen - Nürnberg and Technical University München. Works since 1978 as Systems Analyst, Project Manager and specialist for methods and tools of software development. Since 1997 consultant at CMG, specialised in project management and consulting for CMG's TestFRAME ® testing methodology.

CMG



Testing of e- commerce systems

Requirements and solutions

Gunthard Anderer, CMG

CMG

Introduction

- The speaker:
  - Gunthard Anderer
  - Senior consultant for testing and project manager in various CMG TestFrame ™ projects
  - e-mail: gunthard.anderer@cmg.de
- CMG
  - Independent global information and communication technology group
  - ca. 12.000 employees worldwide
  - more information: www.cmg.com

CMG

Agenda

- Do e-commerce systems need more testing than traditional IT - systems ?
- Technical structure and testing requirements of Crazy-T-Shirts.com
- How to automate testing ?
- Partners, tools and toys
- The disaster of success

CMG

Some differences

Issue	E-commerce system	Traditional IT- system
Visibility of faults, errors and malfunctions	immediate, world wide	limited
Back – up via personal	none	usually available
Customer escapes by	mouse click	physically leaving shop etc.
Competitors are	a mouse click away	in other place
System availabilty	24 h x 7 d	partly limited to business hours
Time for error recovery	on the run = none	out of business hours

CMG

What does this mean ?

- E - Commerce systems require much higher quality !
- Quality has to be maintained 24 h x 7d !
- Automated Testing is a real requirement !

CMG

Example: Crazy-T-shirts.com

Crazy-T-Shirts.com is a e-commerce business, where you can buy T-shirts with funny pictures printed on them.

As a customer you can choose different quantities, prices, sizes and colours of your shirt.

The picture on the T-shirt can be choosen among 2000 pictures in a gallery, which are displayed as thumbnails and can be enlarged on mouseclick.

If you want, you can send us a picture of your choice as graphics - file and we print it on your shirt.

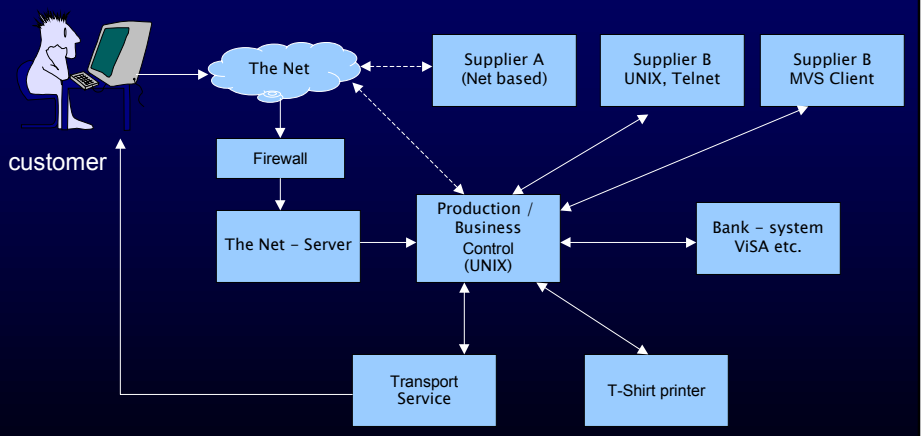
Payment is to be made via credit - card - a secure page for payment is provided.

Delivery is by UPS.

The plain T-shirts are supplied by different suppliers .

CMG

Technical structure of Crazy-T-Shirts.com



CMG

Some observations

- Important is what the customer sees !
- Partly its testing web - pages !
- Its not only testing web pages !
- Performance bottlenecks are likely to be caused by legacy or 3rd party systems !
- We do not know the influence of „ the net“ on performance !
- We have to test components on different platforms .
- Wrong data on the web page can cause a lot of damage !
- Accessability and performance can only be tested with the running, productive system !

CMG

Issues to be tested

- Web - Page
  - Broken links ?
  - All frames correctly displayed on various browsers ?
  - Graphics displayed properly ?
- Performance
  - Do pages build up in due time (8 sec) ?
  - How is the response time, when features of „production / business control“ or 3rd party systems are required ?
- 3rd party interfaces
  - Correct data delivered and received ?
  - Correct function (with our parameters and data)
  - Availability
- Business process
  - Does the customer get what he ordered and do all other processes work as expected ?

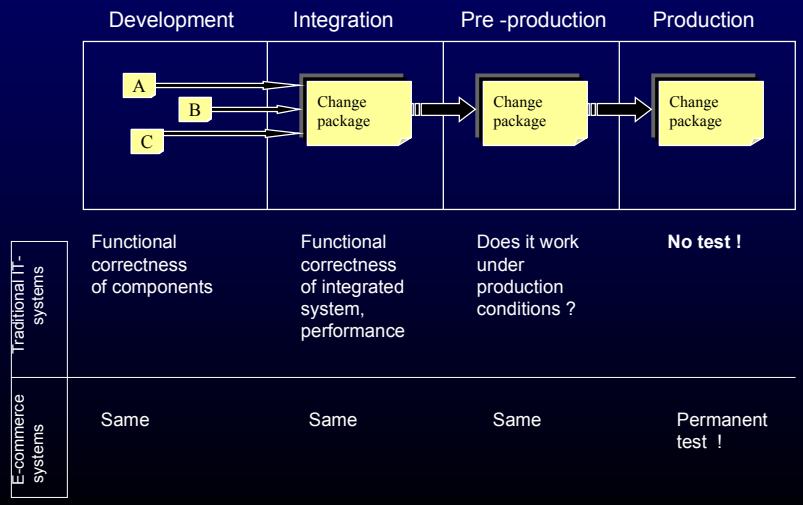
CMG

Stages of testing an e-commerce system

- Before production
  - Web - pages (complete)
  - Components
  - Integration of subsystems / 3rd party systems
  - Data propagation
  - Business process oriented
  - Basic performance tests
- In production
  - Performance
  - Accessibility
  - Business process (random test)
  - Data propagation (random test)

CMG

Test environments



CMG

Example Testprocess

Front - end

- Go to the homepage of Crazy-T-Shirts.com
- Select a certain picture
- Select quantity, price, size and colour
- Key in Credit - Card Data
- Check order confirmation

Back - end

- Check bank account - payment booked ?
- Check invoice - printed and ok ?
- Labels for UPS printed ?
- Shirt produced (feedback from shirt - printer there) ?
- If stock of plain T-Shirts low: Supplement order executed and ok ?
- Did UPS deliver in time ? Check in UPS - system !

CMG

How to automate the testprocess ?

TESTFRAME

CMG's integrated method for:
test development
test automation
test organization

CMG

How does TestFrame work ?

- test development aimed at the production of "clusters"
  - input and expected results
  - test language with "action words"
  - in spreadsheets
- automatic execution by a "navigation script"
  - written in the script language of the cast tool
  - general part: the engine
  - specific part: the action words

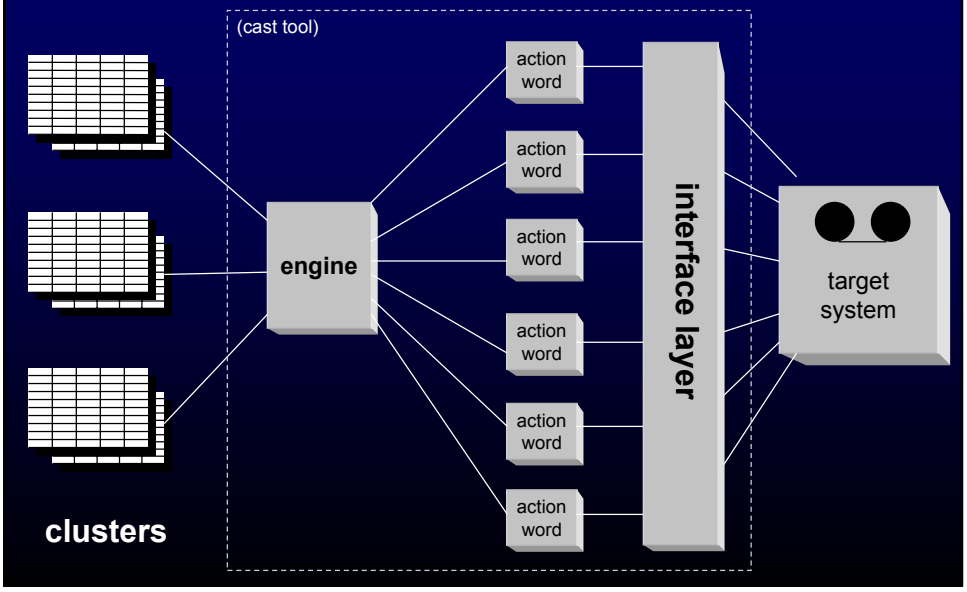
CMG

Logical test description - the cluster

cluster	Crazy-T-Shirts_01					printonlyfaile
author	Gunthard Anderer					
version	0.1					
date	20.08.2000					
	Browser					
Start	IE					
testcase	**C01T001**	**The Webside - Test**				
	URL					
Select page	crazytshirts.com/pictures					
	picture name	*reference bitmap*				
Select picture	mypicture.jpg	mp001				
	quantity	*size*	*price*	*currency*	*name*	*adress*
Input order	2	XL	10	EUR	G.Anderer	Munich
	number	*Holder*	*valid*	*company*		
creditcard data	1234 2453 2667 6345	Test user01	01/01	VISA		
	ordernumber					
check confirmation	&keep[ordernumber01]					
testcase	**C01T002**	**Test the back-end**				
	Ordernumber					
Check bank account	&ordernumber01					
check invoice	&ordernumber01					
etc.						

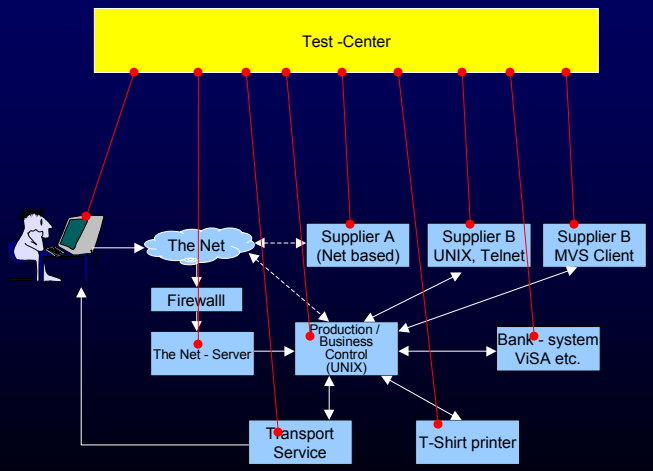
CMG

Layout of the navigation



CMG

Integration of tools and control



CMG

The winds of change

- Values change over time - your cluster has to change with them
- You need data of the system under test at run time
- Tests consist of several parts and have to run in discontinuous time frames
- You have to generate test input
- You have a lot of different target systems
- Functions and processes in the target systems change
- 3rd party systems can only be tested in production

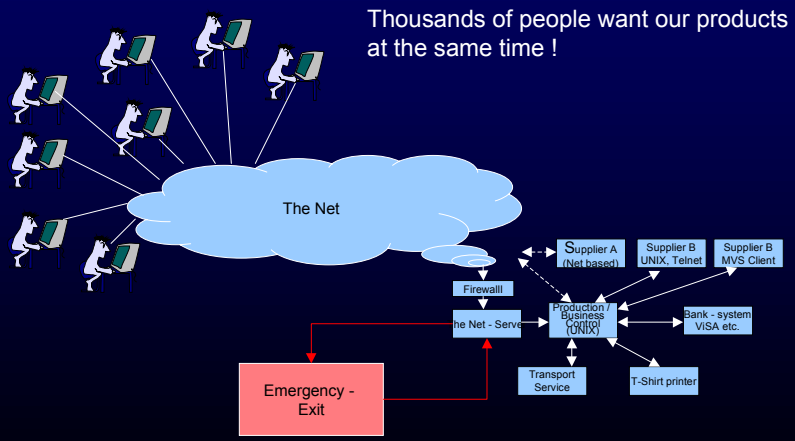
CMG

Partners - what for ?

- Set up your testing system with a partner
  - Don‘t invent the wheel another time !
  - Save time and money !
- Let your tests via „the net“ be executed by a partner
  - Don‘t invest in infrastructure not much used !
  - Tests from all over the world are too expensive !

CMG

The disaster of success



QWE2000 Session 8I

Mr. Eric Messin [USA]
(Vality Technology)

"Ensuring Data Quality for E- Commerce Systems"

Key Points

- Why data entered through e-commerce is particularly prone to error
- How erroneous e-commerce data can wreak havoc on back-end databases
- How faulty e-commerce data diminishes the accuracy of front- AND back-end business transactions
- A high-level outline of technology solutions that address data quality in e-commerce
- How such solutions can enhance e-commerce ROI and overall business success

Presentation Abstract

Today's cutting-edge businesses are launching e-commerce initiatives with an eye towards working smarter, better, and faster. However, what needs to be addressed first is a severe data quality problem in data entered through e-commerce that puts front-end as well as back-end operations in jeopardy. This presentation will lay out the imperatives of data quality in e-commerce, and outline the market for corresponding technology solutions.

About the Speaker

As Sales Director of SouthWest Europe, Eric Messin spearheads all Vality sales activities in that region - leading strategic sales and accounts, setting sales direction, and guiding and developing the SouthWest European region sales team and pre- and post-sales activities.

Mr. Messin has extensive business experience in sales and management positions. Prior to joining Vality in 1999, Mr. Messin was Strategic Alliances Director at Business Objects. Prior to this, he worked at IBM Europe, where he managed key alliances in their Software organization, and IBM France, where he was in charge of business development in trading room activities of the Financial sector. Mr. Messin has also worked internationally, including a one-year stint in Japan and a position at the Banque Indosuez (acquired by Credit Agricole) in New York.

Mr. Messin has spoken on the topic of business intelligence at such events as E-business Intelligence Solutions (1997), Business Intelligence World (1998) and IBM solutions (1999). He holds an MBA in Corporate Finance from Paris Dauphine

University.

Ensuring Data Quality for E-Commerce Systems

The Hidden Role of Data Quality in E-Commerce Success

Eric Messin
Director of Sales
La Grande Arche Paroi Nord
92444 Paris La Defense
France
011-33-140-90-3518
www.vality.com

Quality Week Europe 2000
November 22-24, 2000

E-Commerce: Fast Track To The Promised Land?

- Electronic highway's benefits
  - Instant, global, two-way communication
    - More buyers
    - Seamless ties to partners, suppliers, customers
- Potholes in the electronic highway
  - Easier, "unattended" access makes it more difficult to provide customer service
  - Increased volume of data = increased data variations, misinterpretations and misrepresentations

Quality Week Europe 2000
November 22-24, 2000

Obvious Challenges

- Shoppers' product searches garner
  - Too many irrelevant results; or
  - No results at all
- Customers' free-form data entries not easily linked to back-end standards
- Internal systems' data "not ready for prime time"

E-Commerce: Fast Track To The Promised Land?

- The answer is a qualified "Yes"
- The key enabler is data quality
- E-commerce requires a common, data quality solution – e-quality content management

Defining Data Quality In The E-Commerce Context

- Data quality misconceptions
  - Data is either right or wrong
  - Data is either correctly or incorrectly represented
- Misconceptions oversimplify the fact that perception of data quality varies by user and usage

Data Quality Demystified: An Example

Name	*Street*	*City*	*Zip*
DEC	216B Old County Rd	Lincoln	01773-4603
Digital Equipment Corp	216 Old County Rd Ste B	Lincoln	01773-4603
Compaq	216 Old County Rd	Lincoln	01773-4603

- Order fulfillment, shipping, direct mail ✓
- Marketing ✘
- Contract administration ✘

Data Quality Demystified

- Data quality not an absolute state
  - It is a relative assessment of the degree of usefulness and reliability for an intended purpose
- The more user communities and business processes served by the same data, the greater the risk

Data Quality Is Multidimensional

- Data quality has a time dimension
  - Data values that are correct when recorded may become liabilities in the future
    - Marriages
    - Mergers
    - Zip code reassignments
    - Area code expansions
  - Historical representations often have intrinsic and legal "correctness" that must be preserved

Data Quality Is Multidimensional

- Data quality is contextual
  - Data in internal systems may be inappropriate to expose to a wide public
    - Special discounts and negotiated terms
    - Confidentiality
    - Internal comments
      - Data collected to assist delivery personnel may alienate customers if exposed – "Mean dog in back yard", "stone deaf: holler at them"

Data Denial

- Data quality problems have always existed
- Problems have been
  - Masked or managed by business logic of legacy applications
  - Hidden by an intermediary
- E-commerce "pierces the veil" of operational data

E-Commerce Trends Accentuating Data Quality Issues

- Breadth of user exposure
- Diversity and volume of source data
- Disintermediation
- Privacy and ethical issues

Breadth Of User Exposure

- E-commerce is extending the exposure of the relative nature of data quality
- Older operational data may be unclear or inappropriate, particularly to user whose perspective may be different than anticipated

Example: 14-gauge GRN cable

Is it 14-gauge green cable or grounded cable?

Diversity And Volume Of Source Data

- Transactions must be quick and convenient
  - Minimum of edit checking and verification
  - Web data not easily reconciled with operational data for analyses
    - May even deliberately circumvent fraud and abuse of purchasing policies

Example: Steve Brown, S Brown, Stephen Brown, Steve Bron – Same customer? On credit hold?

Disintermediation

- E-commerce has a self-serve business model
  - Positives: Speeds transactions, saves money
  - Negatives: Loss of buffer or translator, discrepancies left unresolved, loss of revenue

Example: I search for pants, you sell slacks. An intermediary would automatically make the translation and the sale – not any more!

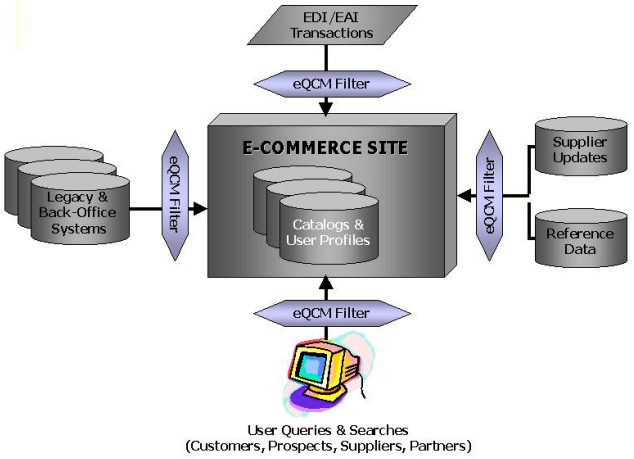
Privacy And Ethical Issues

- Misuse of private data is gaining increasing publicity
- Not just a legislative issue – it's also a data quality issue
  - Incorrect social security # reveals information
  - Inability to identify unique customer reveals financial data

The E-Quality Content Management ("eQCM") Process

- When does it occur?
  - Anytime that data is in motion
  - Not a one-time "clean-up"
  - Real-time data quality maintenance filters are required
  - eQCM must mediate the daily activity that goes against your enterprise databases

eQuality Content Management Control Points



What Processes Are Involved?

- Context mediation
  - Determination of business meaning of word or value based on context
- Fuzzy searching
  - Ability to perform quick and product retrieval of data without a precise key
- Fuzzy matching
  - Application of your business rules, and measure of statistical certainty, to determine when to declare critical relationships within the data, such as duplication and affiliation

What Does It Cost?

- Costs vary depending on the complexity of the data and affected business processes
  - Costs escalate in proportion to amount of risk/reward
    - High-dollar transactions necessitate very high data quality
- *WARNING*: One size does not fit all
  - For example, shrink-wrapped "data-cleansers" designed to prepare bulk mailings will not provide data quality for sales, marketing and CRM efforts

Conclusion

- You must have a requirement to manage data quality whenever:
  - Data is re-purposed or deployed for new business purposes
  - External data sources are integrated, linked or queried
  - Transactions of external origin are synchronized with your Web-site or back office
  - Users (prospects, customers, partners) search your published databases

Ensuring Data Quality for E-Commerce Systems

The Hidden Role of Data Quality in E-Commerce Success

Eric Messin
Director of Sales
La Grande Arche Paroi Nord
92444 Paris La Defense
France
011-33-140-90-3518
www.vality.com

Quality Week Europe 2000
November 22-24, 2000

QWE2000 Session 8M

Mr. Martin S. Feather, Mr. Tim Kurtz [USA]
(NASA Glenn Research Center)

"Putting It All Together: Software Planning, Estimating And Assessment For A Successful Project"

Key Points

- Software Estimation
- Project Planning
- Assessment, Risk Mitigation

Presentation Abstract

OBJECTIVE

How can project managers best incorporate estimation, planning, risk management, corporate processes and resources into a coordinated, tailored approach to developing and managing software? Large software efforts face the challenges of (1) identifying software risks, (2) judiciously planning for IV&V, development & QA activities to mitigate risks, (3) estimating cost & schedules of these plans, (4) accommodating the priorities of the primary stakeholders while ensuring mission success, (5) complying with standards & processes. This presentation will describe and demonstrate a coherent approach and accompanying tool support that addresses these challenges.

APPROACH

The first phase of the approach uses a user-friendly electronic interview to elicit overall project characteristics. Behind the scenes, relevant portions of this data are automatically fed into COCOMO II to yield cost and schedule estimates. The COCOMO estimates, and other portions of the user-provided data, are then combined with institutional practices and policies (for example, ISO 9001, CMM and site-specific principles), yielding estimates of cost, schedule, and risk, and plans for mitigating risk while conducting software development.

The second phase of the approach allows the user to scrutinize and tailor this development plan. Key to this phase is the use of several cogent visualizations of risks, mitigations, and their interrelationships. Through these visualizations the user can both comprehend and customize the plan and its impacts (on cost, schedule, and risk). The final phase of the approach combines all of the above to yield the

outputs of this whole approach, namely:

- Identified software risks. The risk lists will take into consideration risks associated with software failures on previous NASA and aerospace missions (lessons learned, failure reports, defect profiles, etc.). This includes the identification of software components and intermediate deliverables by level of system criticality.
- An optimized plan that identifies software IV&V, development, and QA activities that mitigate and eliminate software risk for a given project at various times during the lifecycle.
- Consistent cost and schedule risk reduction budget estimates that establish a responsible balance between constrained project funding and the safe implementation of software subsystems.
- An equitably negotiated IV&V, Software Development, and QA plan that includes the priorities of the primary stakeholders while maintaining a high integrity program.
- IV&V, QA, and project plans that are compliant with institutional policies, ISO based Software Development Process Descriptions, and best practices from (for example) CMM.

CONCLUSIONS

The elements of the approach have been successfully developed and applied separately, in the form of NASA Glenn's AskPete tool (http://tkurtz.grc.nasa.gov/PETE/Default.htm) and NASA JPL's DDP tool ("Scalable Mechanisms for Requirements Interaction Management", Feather et al, in Proceedings IEEE 4th Int. Conf. on Requirements Engineering, IEEE Computer Society, 2000). Determination of the value of the combined approach is currently in process. The added utility of a planning tool that combines estimation, tailorable development processes, QA and IV&V estimates, and risk management based on the integration of these disparate but related activities is believed to provide a definite benefit to program managers.

About the Speaker

Tim Kurtz:

For the 13 years prior to joining Scientific Applications International Corporation (SAIC), Tim worked for Defense Contract Management Command (DCMC). During that time, he was the program manager for the Mk 48 ADCAP torpedo program at DPRO Westinghouse and implemented the software quality assurance program and monitored the transfer of software and development of test equipment for the Mk 50. At DCMC Dayton he was responsible for training and overseeing the SQA activities of Software Quality Assurance Specialists who monitored DoD software development contracts and the development and maintenance of all Air Force simulators. Trained in ISO 9000 auditing and Software Development Capability Evaluation Training, Tim developed and implemented the ISO 9000 Qualification Audit system for DCMC Dayton to provide second party ISO certification to defense

contractors and provided software certification training for all Software Professional Development Program applicants in DCMC.

Martin S. Feather:

For the last four years, Senior Engineer in NASA/JPL's Software Quality Assurance Group. Prior to that, was a research scientist at Univ. of Southern California's Information Sciences Institute for 15 years, and a research scientist with Principal Investigator status on NSF and DARPA tasks at Computing Services Support Solutions (Los Angeles) for 3 years. Holds BA and MA degrees in Mathematics and Computer Science from Cambridge Univ., England, and PhD degree in Artificial Intelligence from the Univ. of Edinburgh, Scotland. Has over 50 refereed publications in the areas of automated support for early-phase lifecycle software development, and is an active participant in the software engineering milieu.

Putting It All Together: Software Planning, Estimation and Risk Assessment for a Successful Project

Tim Kurtz
SAIC
NASA Glenn Research Center
Cleveland, OH, USA
http://tkurtz.grc.nasa.gov/pete

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Quality Week 2000
Brussels, Belgium
November 23, 2000

Research funded by NASA OSMA and GSFC IV&V Facility

Approach to Putting "it" Together

Three phased approach

- Characterize the project
  - COCOMO II
  - ISO 9001 Development Processes
  - IV&V Criteria
- Tune the Risk Strategy
- Planning and Implementation

Phase 1: Characterize the Project
Use Common and Specific Tools

- COCOMO Cost Estimation
- Software Control Level Matrix (tailored version of DERA Size matrix)
- NASA Independent Verification and Validation Criteria

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 1: Characterize the Project
Development Framework - COCOMO II

Factors derived from COCOMO II

- Cost of development
- Schedule
- Personnel requirements
- Size of project
- Software Reuse

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 1: Characterize the Project
Development Framework - Control Level

Characterization Factors from Control Level

- Organizational Complexity
  - Customers (internal – multiple industries)
  - Development site(s) (single – multiple sites)
- Technical Complexity
  - Degree of Innovation
  - Use of tools
  - Interdependencies of Deliverables
- Consequence of Failure
  - Safety Implications
  - Business Implications

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 1: Characterize the Project
Determine Need for IV&V or Independent Assessment

Factors From Independent Verification & Validation

- The need for IV&V is based on possible effect and extent of failure of the software to perform as intended.

Factors Used to Calculate:

- Resources (manpower) expended
- Investment (money) expended
- Effect of failure on personnel and equipment

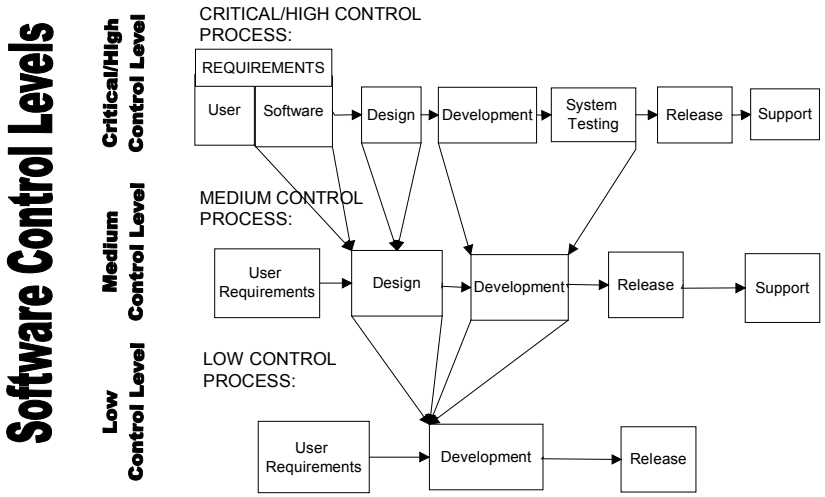
Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 1: Characterize the Project

Tailor Development Processes



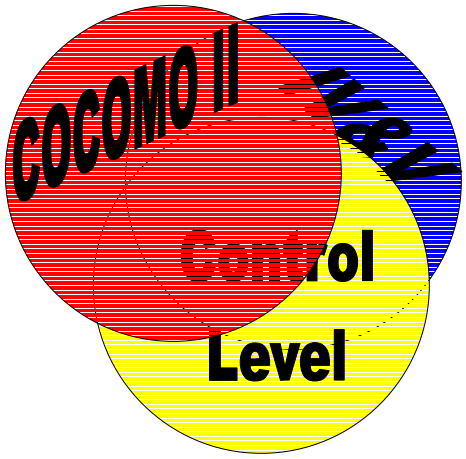
Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 1: Characterize the Project

Build on Common Areas



- COCOMO II factors address majority of the development planning issues
- Control Level factors overlap COCOMO II and address additional organizational and performance issues
- Incorporating other areas of interest, (i.e. IV&V, Software Assurance), build on COCOMO II and Control level questions

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 1: Characterize the Project

Outcomes

- What the project will cost (personnel, money)
- How long the project will take
- What controls are required, What documents need to be generated
- What activities need to be performed
- An initial set of risk mitigations based on the project's parameters

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 1: Characterize the Project

Identify Initial Mitigation Set Based on Control Level

Pac tId	Pact Title	Percent Time	Percent Cost	Absolute Cost	Absolute Time	Low Pact	Medium Pact	High Pact	Critical Pact
P9	Requirements	0	0	$0.00	0	X	X	X	X
P10	Authorization to proceed	0	0	$0.00	0	X	X	X	X
P11	Identify design/coding standards	0	0	$0.00	0	X	X	X	X
P12	Maintain Softw are Development Folder	0	0	$0.00	0		X	X	X
P13	Softw are Assurance review s Management Plan	0	0	$0.00	0		X	X	X
P14	Implement Problem report and corrective action system	0	0	$0.00	0		X	X	X
P15	Management Plan approval	0	0	$0.00	0	X	X	X	X
P16	Documented requirements	0	0	$0.00	0	X	X	X	X
P17	Peer review of requirements	0	0	$0.00	0		X	X	X
P18	Conduct formal inspection of requirements	0	0	$0.00	0				X
P19	Softw are Assurance review s requirements	0	0	$0.00	0			X	X
P20	Requirements approval	0	0	$0.00	0	X	X	X	X
P21	Peer review of plans	0	0	$0.00	0			X	X
P22	Implement Formal configuration management	0	0	$0.00	0			X	X
P23	Conduct Product Assurance Audits	0	0	$0.00	0			X	X
P24	Conduct Formal Review s	0	0	$0.00	0			X	X
P25	Document approval of requirements and formal review	0	0	$0.00	0			X	X
P26	Customer approval of certification procedures	0	0	$0.00	0				X
P27	Conduct analyses of criticality and safety	0	0	$0.00	0				X
P28	Plan and schedule IV&V activities	0	0	$0.00	0				X
P29	Identify method for verification of safety critical functions and requirements	0	0	$0.00	0				X

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 2: Tune Risk Strategy
Terminology

- Risks - combination of likelihood (probability of occurrence) and impact (how much damage it will do if it occurs).
- PACT - risk mitigation, implementation of a PACT will have some effectiveness in reducing one or more risks
  - **P**reventive measures
  - **A**nalyses
  - process **C**ontrols
  - **T**ests

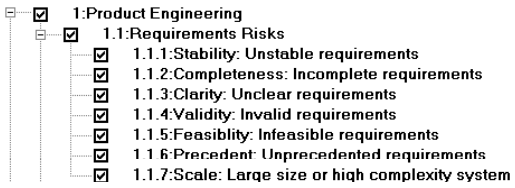
Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 2: Tune Risk Strategy
Identify and Prioritize Specific Risks

- Identify project specific risks
  - Start with the Software Engineering Institute's Software Risk Taxonomy as the base set of software development risks
  - Remove inapplicable risks and add project specific risks



Prioritize risks based on requirements

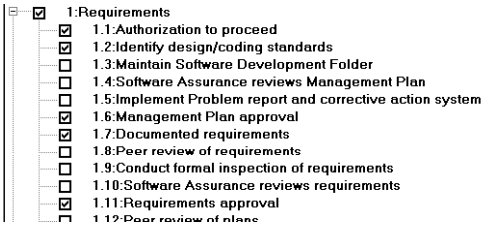
Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 2: Tune Risk Strategy

Identify Risk Mitigations



- An initial set of mitigations was identified based on the characteristics of the project
- Adjust the mitigations based on available resources and impact to the risks

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 2: Tune Risk Strategy

Estimate Risk Mitigation Effectiveness

PACTxFM — Col = Stability: Unstable requirements; Row = Authorization to proceed

		FMs	[-]Product Engineering								
		FMs	[-]Requirements Risks							[-]Design Ri	
		FMs	Stabilit	Compl	Clarity	Validity	Feasib	Precec	Scale:	Functic	Diffic
PACTs	PACTs	FoM\R	0.2835	0.0405	0.0405	0.3645	0.2835	0.2205	0.2657	0.0255	0.02(
[-]Requ	Authori	7.95	0.1	0.1	0.1	0.1	0.1	0.3	0.1		
	Identify	2.5								0.3	0.3
	Mainta	0									
	Softwa	2.65									
	Implen	1.85	0.9	0.3	0.9	0.9	0.3	0.3	0.1		
	Manag	0.15									
	Docum	1.7	0.3	0.9	0.9	0.1	0.3	0.3	0.1	0.1	
	Peer	2.85	0.9	0.9	0.9	0.9	0.9	0.9	0.1	0.1	
	Condu	2.85	0.9	0.9	0.9	0.9	0.9	0.9	0.1	0.1	
	Softwa	2.8	0.9	0.9	0.9	0.9	0.9	0.9	0.1	0.1	

- Each mitigation only affects a subset of risks
- Each risk is affected to a different degree
- The effect of a mitigation on a risk may need to be adjusted from one project to another

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 2: Tune Risk Strategy

Tune Mitigations to Maximize Resources

Risks	Mitigations						
1.4.2	☒ 3.2	☐ 3.3	☐ 1.19	☐ 2.12	☒ 4.1	☐ 5.1	☐ 5.3
2.1.2	☒ 1.1	☐ 3.4	☐ 1.19	☐ 3.3			
1.4.1	☒ 3.2	☐ 1.19	☐ 2.12	☐ 3.3	☐ 3.7	☐ 3.8	☒ 4.1
	☐ 4.2	☐ 4.5	☐ 4.6	☐ 3.10			

- Select mitigations
  - Which have a greater impact on a single risk
  - Affect a range of risks

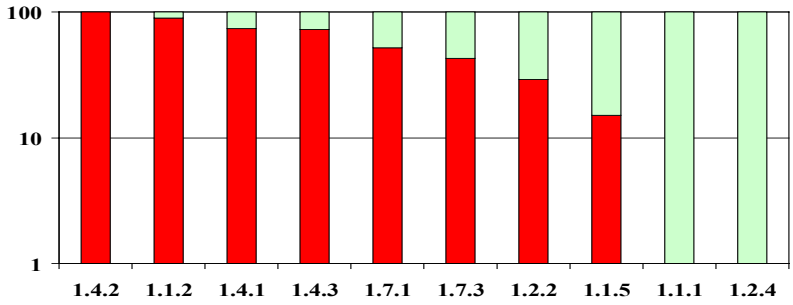
Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 2: Tune Risk Strategy

Tune Mitigations to Maximize Resources



Pareto risks to determine which have yet to be mitigated to an acceptable level

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 2: Tune Risk Strategy

Outcomes

A tailored set of risk mitigations for the project which includes

- A set of risks applicable to the project
- A set of risk mitigations applicable to the project risks
- The costs of the risks and mitigations in time and effort

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 3: Planning and Implementation

Combine and Implement Mitigations

- Original estimates are supplemented based on selected mitigation strategies
- Resulting impact on the project can be reviewed and adjusted
  - Budget
  - Schedule
  - SPA activities

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 3: Planning and Implementation
Develop Plans

- Project Development and Product Assurance plans are developed based on
  - Characteristics of the project
  - Risks
  - Risk mitigations
  - Organizations development activities
  - ISO 9001
  - COCOMO II estimates

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 3: Planning and Implementation
Development Plan

- Start with a development plan for a Critical Control project
- Tailor the plan to the appropriate level by removing activities and deliverables that don't provide the needed cost/benefit ratio for the effort.
- Address each development phase and associated documentation

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 3: Planning and Implementation
Other Project Tasks

Software Product Assurance Plan

- Lists needed Software Product Assurance activities for the level of control
- Provides Software Product Assurance effort estimates based on the results

Level of IV&V

- Identifies if Independent Assessment or IV&V necessary for the project
- Suggests activities and processes
- If IV&V is indicated: level and tasks should be negotiated and documented in Software Management Plan

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Phase 3: Planning and Implementation
Outcomes

- Development Plan based on:
  - control level
  - documentation requirements
  - risk
- Product Assurance Plan based on:
  - development activities
- Risk mitigation activities based on:
  - control level
  - risks
  - resources

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Summary for Putting It All Together

- Provided approach for managing software development
- Described a process
  - Coordination between factors
  - Tailoring to specific project needs
- Presented a Framework, incorporating
  - Estimation
  - Corporate Processes and Resources
  - Risk Management
  - Planning

Tim Kurtz
SAIC/NASA Glenn Research Center
Cleveland, OH, USA

Martin Feather
NASA Jet Propulsion Lab
Pasadena, CA, USA

Research funded by NASA OSMA and GSFC IV&V Facility

Putting it All Together: Software Planning, Estimating and Assessment for a Successful Project

Tim Kurtz
Science Applications
International Corporation,
NASA Glenn Research Center
21000 Brookpark Rd, MS 501-4
Cleveland, Ohio 44135, USA
Tim@bfsng.com
http://tkurtz.grc.nasa.gov/pete

Martin S. Feather
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109, USA
Martin.S.Feather@Jpl.Nasa.Gov

Objective

How can project managers best incorporate estimation, planning, risk management, corporate processes and resources into a coordinated, tailored approach to developing and managing software? Large software efforts face the challenges of (1) identifying software risks, (2) judiciously planning for IV&V, development & QA activities to mitigate risks, (3) estimating cost & schedules of these plans, (4) accommodating the priorities of the primary stakeholders while ensuring mission success, (5) complying with standards & processes. This paper describes a coherent approach and accompanying tool support that addresses these challenges. The approach consists of three phases. The first phase characterizes the project, the second phase balances the risks in the project with the resources available to mitigate them and the last phase combines the results into plans for controlling the project. Two tools, Ask Pete, developed at NASA Glenn Research Center and DDP, developed at the NASA Jet Propulsion Lab, combine to assist project managers in completing the activities in these phases.

RESEARCH FUNDED BY NASA OSMA AND GSFC IV&V FACILITY

Characterize the Project

The first phase of the approach uses a user-friendly electronic interview to elicit overall project characteristics. Behind the scenes, relevant portions of this data are automatically fed into COCOMO II to yield cost and schedule estimates, and into other matrices to yield control level determinations and IV&V estimates. The COCOMO estimates, and other portions of the user-provided data, are then combined with institutional practices and policies (for example, ISO 9001, CMM and site-specific principles), yielding estimates of cost, schedule, and risk; and plans for the development effort, oversight and risk mitigation while conducting software development.

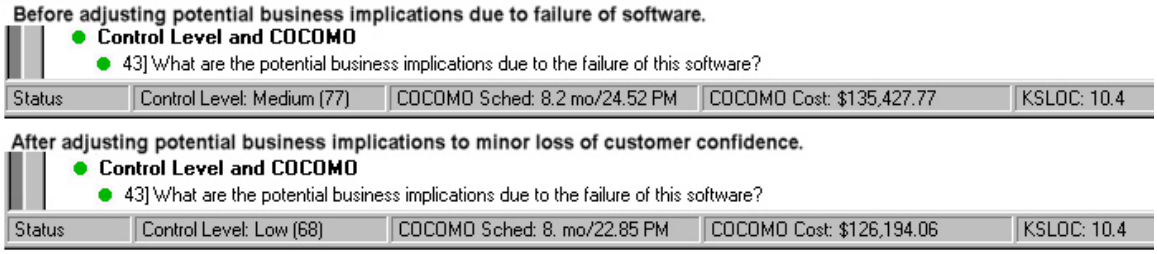


Figure 1, Change to Control Level Shown in Status Bar

Estimate Cost and Schedule

Determine the SLOC estimate using COCOMO II to scale the level of effort and develop an estimate of the cost and schedule based on what is known about the project, personnel, organization and resources. The COCOMO II questions form the foundation for this phase, providing much of the information needed for the control level and IV&V effort determinations. These initial schedule estimates form the basis for planning the project. The responses can be tuned, to a degree, to adjust the time and resources required to complete the project. Tradeoffs and the effects of software reuse can be examined by monitoring the Ask Pete status bar, see Figure 1.

Determine the Development Framework

Using the matrix in Table 1, identify the risks in three areas. These areas lay the foundation for identifying the level of control that must be implemented to minimize risk while optimizing the use of resources and maximizing the likelihood of successful project completion.

Identify Operational Risks

Identify the operational risks to the equipment, and personnel and the mission if the software should fail or be degraded. These risks can directly impact the chances for mission success.

Operational risks deal mainly with what will happen if the software fails during its functional use. Identify effects of the failure on personnel, the platform the software controls or operates as well as other equipment which may be negatively affected by adverse behavior and the ability for the mission to be successfully completed. As shown in the accompanying matrix, these risks can be large enough to require the imposition of Critical Control on the project irrespective of the magnitude of the other risks.

• Table 1, Control Level Matrix

Factor	1	2	3	4	5
Resourcing					
Software cost annualized over life of project (including all civil servants and contractors)	$100K - $500K	$500K – 1M	$1M - $2M	$2M - $20M	$20M and up
Organizational Complexity					
Project development location	Own Group	Several Groups, most at GRC	More than 2 other sites	More than 3 other sites	Numerous sites
Customers	Self	Other in own Directorate or one customer group, low number of users	Within GRC	Within NASA	Entire Industry or multiple industries
Developers Software experience level	All team qualified and experienced	Most team members qualified and experienced	Half the team qualified and experienced	Few team members experienced	Experienced staff not available
Technical Complexity					
Test Risk	No testing required	Minimum Testing	Standard testing required	Integrated Testing	Major testing effort required (e.g. IV&V)
Degree of Innovation	Well proven, known to GRC	Proven with some GRC experience	Proven, but new to GRC	Partially proven with some pioneering	Pioneering
Software development tool availability	All software development tools already purchased/in-house/familiar with	Majority of the software development tools are purchased /in-house/familiar with	Software development tools must be identified and purchased and learned	Majority of software development tools must be obtained, remainder developed	All software development tools must be developed
Interdependencies of deliverables	Simple standalone	Some Integration	Integrated	Highly integrated	Fully integrated
Safety Implications * (see NASA STD 8719.13A, NASA GB-1740.13 & NHB 1700.1 (V1-B))					
Potential Damage to carrier vehicle, major equipment, or system itself	No damage	Small/minor damage to equipment, or to system itself, mission still possible	Repairable/ recoverable damage to system and little or no damage to any related or surrounding systems	**Loss of system, and/ or damage to any critical surrounding systems or carrier vehicle**	**Loss of carrier vehicle (e.g. space craft, aircraft, major satellite)**
Potential Injury to personnel	No injury	Minor injury	Injury	**Severe injury or temporary disability**	**Loss of life or permanent disability**
Business Implications					
Consequence of Failure	Minor loss of Customer Confidence	Unsatisfied Customer	Damage to GRC Reputation	Damage to NASA reputation	**Significant mpact to USA**
Schedule Pressure	No time pressure	Little effort to meet milestones	Nominal effort to meet milestones	Aggressive effort to meet milestones	Time critical

Critical Control

|------Low control software -----|

|-- Medium control software--|

|---------- High control software ----------|

Identify Organizational Risks

Identify the risks to the development team and organization in successfully completing the project or if the software should fail or otherwise perform adversely.

Organization risks are concerned with the amount of resources committed to the project, the cohesiveness, quality and experience of the development team(s), the exposure the completed project will be subjected to, and the effect on the public's perception of the organization if the software or development effort should fail.

Identify Development Risks

Identify development risks that could affect successful completion of the project. Successful completion is defined as on time, within budget and with the proper functionality.

Development risks include the level of testing or IV&V required, the incorporation and development of new technologies, the availability and use of development tools and the amount of integration which must be achieved with other software or systems.

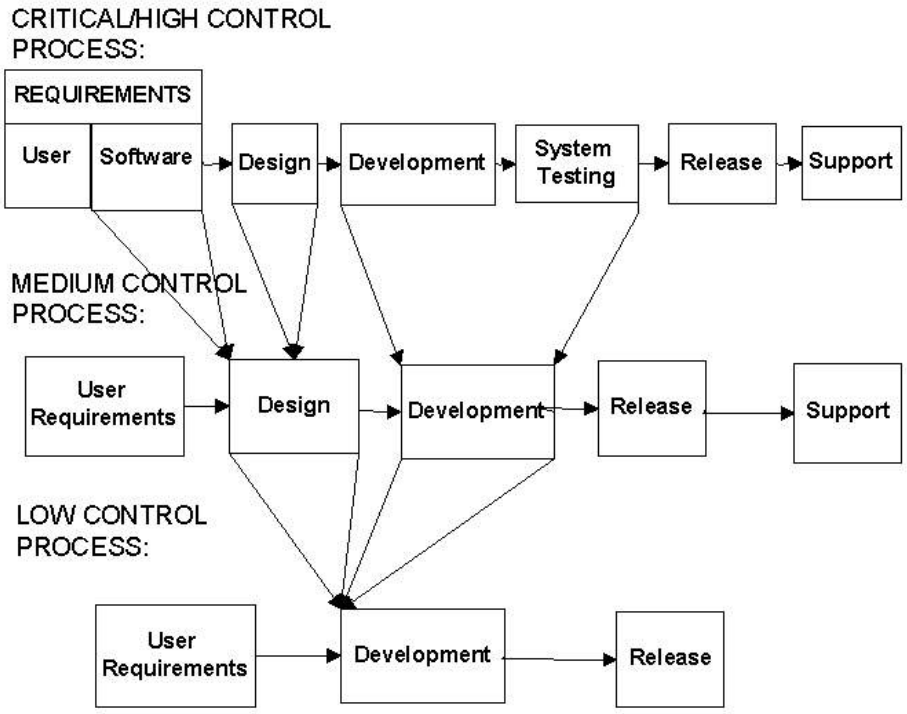


Figure 2, Control Level Phases

Determine Tailoring of Development Practices (ISO 9001, CMM Processes)

Organizations that have implemented ISO 9001 processes need to tailor the processes to the requirements of the project. Smart organizations have a

documented plan for tailoring their activities to fit the scope of a project just as they have a method for tailoring the documentation requirements of the project. Table 2 shows the tailoring of documentation to control level development used with the control level processes in Figure 2

Figure 2 illustrates the tailoring function based on the required control level. Large, complex, risky projects, requiring High or Critical Control are developed with a typical waterfall life cycle approach with requirements, design, development and testing phases. Projects under this control level would typically be characterized by selections from the right two columns of the table. Flight Software for the International Space Station power system or an application to be used across the aeronautics industry would be two good examples.

Medium control levels utilize a development life cycle that includes system testing as part of the Development phase and combines User and Software Requirements Analysis into one activity. The same functions are performed as for a High control project but the phases are not as rigorously defined and controlled and documentation requirements are slightly reduced.

Low control levels combine the Design and Development phases into one. Again the same functions are performed but even less rigorously defined and only minimum documentation requirements are imposed.

Table 2, Documentation Requirement by Control Level

Key: ▲ Required ❒ Optional

Software Documentation for:	Low	Medium	High	Critical
Management Plan	▲	▲	▲	▲
Development Activities Plan	▲	▲	▲	▲
Verification Plan	▲	▲	▲	▲
Validation Plan	▲	▲	▲	▲
Organizational and Technical Interface Descriptions	▲	▲	▲	▲
Acquisition Activities Plan			❒	❒
Training Development Plan			▲	▲
Assurance Plan		▲	▲	▲
Risk Management Plan		❒	▲	▲
Configuration Management Plan		▲	▲	▲
Delivery and Operational Transition Plan			▲	▲
Product Specification			▲	▲
Concept documentation			▲	▲
Requirements documentation	▲	▲	▲	▲
Design documentation	▲	▲	▲	▲
Version description		❒	▲	▲
User's guide	❒	❒	❒	❒
Operational Procedures Manual	❒	❒	❒	❒
Procedures				
Testing Procedures	▲	▲	▲	▲
Safety Assurance Procedures			❒	▲
Security and Privacy Procedures			❒	▲
Certification Procedures		❒	❒	▲

Utilize Results of Previous Steps

Utilizing the estimates and the risks to the project and the organization that have been identified provides a basis for tailoring the development processes that will be followed. Many of the questions answered during the estimation process were also asked again during the control level identification process. By integrating the two activities into one you assure consistency in the results when the two are combined. Use these results as the basis for tailoring the ISO 9001 processes. Be even more efficient by combining them into a single application which provides an initial development cost and schedule estimate, a compilation of processes that will reduce the risks to the project, organization and provide the best path to success, a development plan, a quality assurance plan, schedule and estimate and an IV&V assessment.

Additional decision making matrices can be added to this process, i.e. NASA NPG 2820 contains an appendix for containing the criteria for implementing Independent assessments or IV&V on a project. What you will find when incorporating additional decision matrices to the COCOMO/Control Level foundation described above, is that many of the factors in the new matrix have already been addressed. The IV&V matrix included 15 decisions but only required the addition of four new questions to the interview process and tweaking the existing reports to incorporate the new results.

Adjust Resources and Project

Having done all that, based on what you thought you knew about the project, did it agree with your initial assessment? Now that the data gathering is done, it's time to look at what happened when all the parts were combined. Is it an elephant, a horse or something out of mythology?

If you find that it's not what you initially expected, then the next step is to look at what can be changed to make it more familiar or achievable. If cost or time required to develop is the issue, adjust the COCOMO factors. If the risks are too high or the controls too stringent, adjust the Control Level factors. Too expensive and controlled? Adjust the factors that affect them both first. Once, you've done all that you rationally can, then decide is it something your organization wants to attempt, or should you pass on it. If you decide to continue, document the decisions that you made and be prepared to act on them.

For example, if your project estimate is over the budget allowed and you planned on using a programming team with little experience (6 months) with the intended platform, and the languages and tools being used (1 year). By using a more experienced team with one year experience with the platform and three years experience with the language and tools, the estimated cost for the project will drop from $317,000 to $265,000.

Pac tId	Pact Title	Percent Time	Percent Cost	Absolute Cost	Absolute Time	Low Pact	Medium Pact	High Pact	Critical Pact
P9	Requirements	0	0	$0.00	0	X	X	X	X
P10	Authorization to proceed	0	0	$0.00	0	X	X	X	X
P11	Identify design/coding standards	0	0	$0.00	0	X	X	X	X
P12	Maintain Software Development Folder	0	0	$0.00	0		X	X	X
P13	Software Assurance reviews Management Plan	0	0	$0.00	0		X	X	X
P14	Implement Problem report and corrective action system	0	0	$0.00	0		X	X	X
P15	Management Plan approval	0	0	$0.00	0	X	X	X	X
P16	Documented requirements	0	0	$0.00	0	X	X	X	X
P17	Peer review of requirements	0	0	$0.00	0		X	X	X
P18	Conduct formal inspection of requirements	0	0	$0.00	0				X
P19	Software Assurance reviews requirements	0	0	$0.00	0			X	X
P20	Requirements approval	0	0	$0.00	0	X	X	X	X
P21	Peer review of plans	0	0	$0.00	0			X	X
P22	Implement Formal configuration management	0	0	$0.00	0			X	X
P23	Conduct Product Assurance Audits	0	0	$0.00	0			X	X
P24	Conduct Formal Reviews	0	0	$0.00	0			X	X
P25	Document approval of requirements and formal review	0	0	$0.00	0			X	X
P26	Customer approval of certification procedures	0	0	$0.00	0				X
P27	Conduct analyses of criticality and safety	0	0	$0.00	0				X
P28	Plan and schedule IV&V activities	0	0	$0.00	0				X
P29	Identify method for verification of safety critical functions and requirements	0	0	$0.00	0				X

Table 3, Requirements Phase Risk Mitigations

Create Initial Mitigation Set

Most, if not all, development efforts share the same or similar risks, and luckily, many of the activities performed during development are forms of risk management or mitigation, i.e. requirements reviews, formal inspections, testing, etc.. Using a standard process that tailors your development activities you can easily identify an initial set of mitigation activities for the project as shown in Table 3.

The table includes a list of all possible mitigation activities that could be performed during the Requirements phase. It also identifies which of the activities are required based on the control levels. These activities, selected for the project's control level, then become the initial mitigation set. Associated costs for these activities can also be identified. This will assist when determining if additional activities should be performed based on the cost to implement an activity versus the cost of the risk occurring or the cost of implementing a less expensive activity.

Outcomes of the First Phase

As a result of this phase, you know

- What the project will cost
- How long the project will take
- What controls are required
- What documents need to be generated

- What activities need to be performed
- An initial set of risk mitigations
- Additional risk mitigations which might be selected. If their costs have been identified then you can compare what their cost to the project might be versus the risks they address.

Identify and Mitigate Risks

The second phase of the approach allows the user to scrutinize and tailor the initial risk mitigation plan developed by the first phase.

This second phase brings into play pre-assembled knowledge that relates the risk mitigations of the development plan to known software development risks. Several visualizations of this information enable to user to comprehend the interrelationships between risks and mitigations. Through these visualizations the user can better comprehend the risk landscape, and make a judicious choice of development activities. The main goal of this phase is to develop a risk mitigation plan that minimizes risk subject to the constraints on the software development effort (notably, schedule and cost).

Upon conclusion of this phase, the tailored development is transferred back to the Ask Pete tool for report generation.

In the subsections that follow we describe the main steps of this second phase, notably:

- Identify Risks
- Prioritize Risks
- Identify Risk Mitigations
- Estimate Risk Mitigation Effectiveness
- Select Risk Mitigations

Identify Risks

The tool is pre-populated with a detailed taxonomy of software development risks taken from the Software Engineering Institute (SEI). A snapshot of a fragment of this taxonomy is shown in, Figure 3:

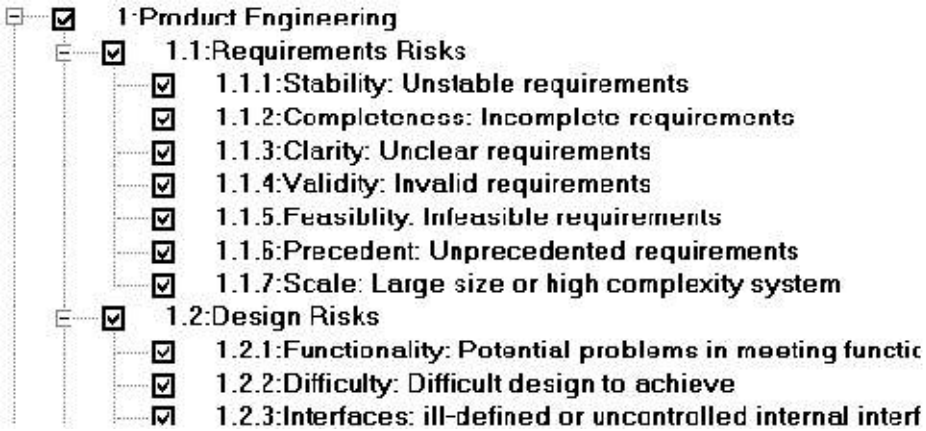


Figure 3, Risk Taxonomy

The user can tailor this risk taxonomy by discarding risks, and by adding new ones.

> Discarding risks: For example, the user might discard a risk that is clearly irrelevant to the project at hand, but should be prepared to justify this action. The tool provides capabilities appropriate to support this practice, by offering a means to switch between views of retained and discarded risks, and by allowing the user to attach commentary to any risk (whether discarded or not). Thus a review team could quickly locate which risks had been discarded by the project team, and scrutinize their justifications for doing so.
>
> Adding new risks: Users can add new risks, presumably risks not present in the SEI risk taxonomy. This enables them to extend the capabilities of the tool to operate on their specific concerns.

This hybrid approach of supplying pre-populated information, together with allowing for user customization, is a recurring theme of this phase. In this particular step of risk identification, the pre-populated risk taxonomy serves as a checklist to remind users of the broad spectrum of risks associated with software development, while the option to add new risks is a means by which users can add risks specific to their task/project/institution.

The net result of this step is a set of software development risks of concern to the task at hand.

Prioritize Risks

It is standard practice to evaluate risk as the combination of likelihood (probability of occurrence) and impact (how much damage it will do if it occurs). Our approach follows this practice – each individual risk has an a-priori likelihood figure (the probability of occurrence were nothing done to inhibit that risk), and separate rating of impact. For rating the impact, we offer a choice between a simple scheme in which the users directly assert their estimations of each risk's total impact, and a more elaborate scheme in which users relate risks to project requirements, and derive the impact from these relationships.

This more elaborate scheme requires further input from the users:

- Identifying the project requirements (which can be individually weighted to reflect their relative importance)

- Quantitatively relating the risks to the requirements that they impact – briefly, the impact of a risk on a requirement is a measure of how much of that requirement would be lost were that risk to occur.

Once these inputs have been provided, the tool automatically computes the total impact of a risk as the sum of the impacts on the weighted requirements.

Advantages of this more elaborate scheme derive from the creation of explicit traceability between risks and requirements: it serves as a disciplined way to make estimates of risk impact, and it results in an understanding of which requirements are most at risk. For example, it is possible to use this information to justify a renegotiation of the requirements themselves, in the case that some of the requirements are seen to be at risk, and mitigation of that risk is particularly expensive. Most importantly, this process allows multiple experts' knowledge to be combined to yield a consistent risk prioritization.

However, elaborating the requirements, and relating them to the extant risks, is a time-consuming process. In the cases where the users already have a good understanding of the relative impacts of the risks, they may prefer to follow the simper scheme and enter those values directly.

The net result of this step is a task-specific prioritization of the previously identified software development risks.

Identify Risk Mitigations

Recall that the first phase of the approach had yielded a set of suggested risk mitigation activities. The second phase presents to the user *all* the possible risk mitigation activities known to the first phase, indicating which of these have been suggested for this particular project. These mitigations, like the risks, are organized into a taxonomy for ease of scrutiny and navigation. In the figure below, suggested activities are indicated by the presence of check marks as shown in Figure 4.

Figure 4, Mitigation List

The user can tailor this mitigation list by discarding mitigations, and adding brand new ones unknown to the first phase. Observe again the hybrid of pre-populated knowledge, in the form of a checklist of known activities, and the option for augmenting this with task-specific additions. Refining the selection from this universe of activities is to be addressed in the last step of this phase.

There are costs associated with performing mitigations. Most notably, budget and schedule are constrained resources in almost all software development efforts, and it is important to know how a suggested set of mitigations will consume these. The first phase of our approach yields cost and schedule estimates for each of the mitigation activities, and these estimates are carried over into this second phase. If users choose to add in additional mitigations, it is their responsibility to provide the costing information.

The net result of this step is a set of risk mitigation activities from which the users can choose.

Estimate Risk Mitigation Effectiveness

Typically, a mitigation will affect only a subset of all the risks, and of those, some more than others. Our approach accommodates this. Mitigations are cross-linked to the risks they mitigate, and associated with those links are quantitative estimates of the effectiveness of those mitigations.

The tool is pre-populated with quantitative effectiveness links between the Ask Pete universe of activities, and risks of the SEI software development risk taxonomy. We have made reasonable estimates of these effectiveness values, which we expect to refine as our experience base grows. Figure 5 shows a fragment of this cross-linking of activities to risks. Activities are the rows, and risks the columns. The numerical values in the white-background cells denote the effectiveness of the activity on the risk; this can range from 0.0 (not effective whatsoever) to 1.0 (fully effective at mitigating that risk). An empty cell is equivalent to an entry of 0.0

PACTxFM Col = Stability: Unstable requirements
Row = Authorization to proceed

		FMs	[-]Product Engineering								
		FMs	[-]Requirements Risks							[-]Design Ri	
		FMs	Stabilit	Compl	Clarity	Validity	Feasib	Precec	Scale:	Functic	Diffi
PACTs	PACTs	FoM\R	0.2835	0.0405	0.0405	0.3645	0.2835	0.2205	0.2657	0.0255	0.02
	Authori	7.95	0.1	0.1	0.1	0.1	0.1	0.3	0.1		
	Identify	2.5								0.3	0.3
	Mainta	0									
	Softwa	2.65									
	Implem	1.85	0.9	0.3	0.9	0.9	0.3	0.3	0.1		
	Manag	0.15									
	Docum	1.7	0.3	0.9	0.9	0.1	0.3	0.3	0.1	0.1	
	Peer	2.85	0.9	0.9	0.9	0.9	0.9	0.9	0.1	0.1	
	Conduc	2.85	0.9	0.9	0.9	0.9	0.9	0.9	0.1	0.1	
	Softwa	2.8	0.9	0.9	0.9	0.9	0.9	0.9	0.1	0.1	
[-]Requ	Requir	0									

Figure 5, Effectiveness Links

The users can adjust the effectiveness values, and annotate them with justifications as they do so. For example, the users might judge that formal inspections will be more effective than the pre-populated data would suggest, because they have an in-house team that is skilled in their application, and past experience has shown their high effectiveness in this area.

If the users have added new risks and/or new mitigation activities, then they will be required to cross-link those new items. For example, if the users add in a mitigation of using model checking (an effective analysis technique that has emerged from the formal methods research community over the last decade or so), they will have to assess which risks it mitigates (e.g., flaws in protocols), and how effective it is at doing so.

The net result of this step is a quantitative cross-linking of mitigations to the risks they address.

Select Mitigations

The primary purpose of this second phase is to allow the users to optimize the suggested program of activities suggested by Ask Pete. The steps up to this point have served to gather much of the key information upon which this optimization is based. However, optimization is not an automated step, rather, the users themselves are expected to explore the options and choose accordingly.

Several visualizations of this information enable to user to comprehend the interrelationships between risks and mitigations. Through these visualizations the user can better navigate the risk landscape, and be supported in making their judicious choice of development activities. A fragment of one of these visualizations follows.

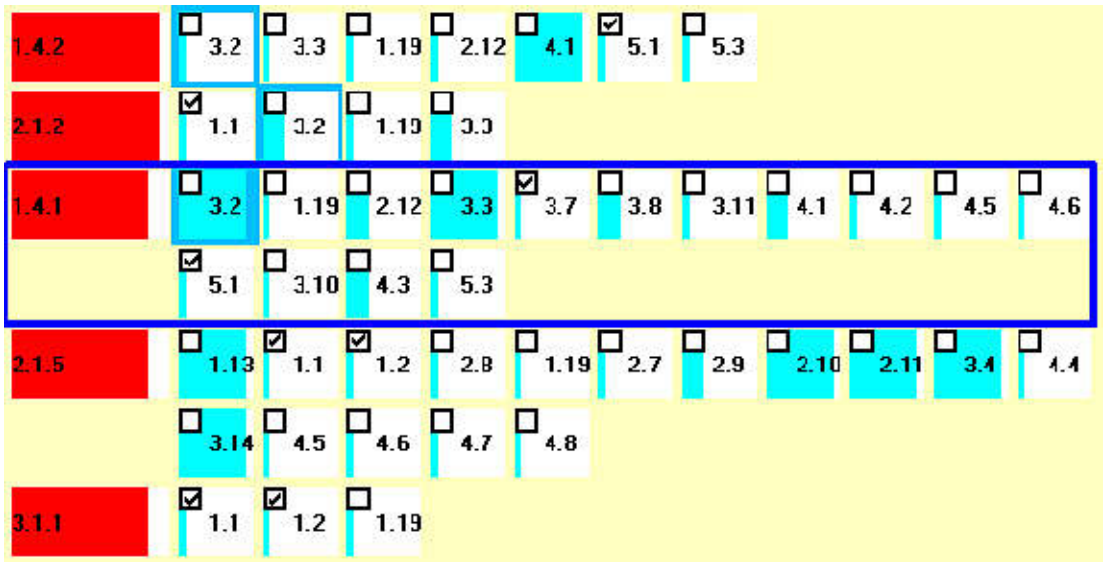


Figure 6, Risks and Mitigation Activities

Figure 6 shows graphically the risks, and for each risk, the activities available to mitigate that risk.

Each risk is displayed as one of the red-colored rectangles in the leftmost column, labeled with the risk number (e.g., 1.4.2) from the corresponding risk taxonomy. The width of the red bar is proportional to the logarithm of the risk. They have been automatically sorted into descending order, hence the width of the red bars diminishes towards the lower part of the figure.

The row to the right of each risk's rectangle displays the activities available to mitigate that risk, labeled with the mitigation number (e.g., 3.2) from the corresponding mitigation taxonomy. If there are too many mitigations to fit into one row, they spill over into a second row, and so on. Each activity has a small check box showing whether or not that activity has been selected, and through which the user can toggle that selection. The width of the turquoise rectangle is proportional to the effectiveness of that mitigation at reducing that risk, for example, mitigation 3.2 alongside risk 1.4.1 (the third row) has a relatively wide rectangle, indicating that it is highly effective at reducing that particular risk; in contrast, 3.2 alongside risk 1.4.2 (the top row) has a relatively narrow rectangle, indicating that it has only a small effect at reducing that risk.

The tool uses a variety of dynamic techniques to provide further detail, highlight selected items of focus (e.g., the border around risk 1.4.1 and its two rows of mitigations), etc.

In general, there is a non-trivial amount of information that users must take into account to make judicious selection of risk-mitigating activities. It does not appear to be feasible to display all this relevant information in one view. Instead, the tool's several visualizations offer a variety of forms of presentation, and a variety of means to focus in on different subsets of the information.

The net result of this step, and therefore of this entire second phase, is the selection of a set of activities together with the costing information associated with those activities. In the course of this second phase, the users could have adjusted the recommended selection that emerged from the first phase in several ways:

- Unselecting originally recommended activities
- Selecting additional activities from base set of mitigation activities
- Adding activities that were not options for the first phase to suggest
- Adjusting the cost and schedule estimates that were made by the first phase.

While the users are making their customizations during this second phase, they may record rationale for these actions. For example, if the users give a risk a relatively low priority, they may wish to record their justification for doing so. Similarly, if they alter the effectiveness and/or cost values of a risk mitigation activity, they again may wish to record a justification (e.g., assert that the cost would be higher than usual because they do not have any staff on hand already trained in that activity, and so would need to expend additional time and budget if they were to apply activity). Such user-supplied justifications are recorded as textual notes, and retained in the database. Users are not required to supply these, but are encouraged to do so both to serve as a reminder to themselves for future reference, and to serve as descriptive rationale to others (e.g., independent revivers of the project plan).

The information of the selected activities and adjusted cost and schedule estimates is transferred to the third phase for generation of various plans and reports, discussed next.

Combine and Implement Mitigations

Once the risk mitigation has been tailored in the second phase, the results replace the initial mitigations from the first phase. New mitigations are added and rejected mitigations are removed. The resulting costs of the mitigations in time and money are totaled and presented with the project estimate and control level information. The results are then incorporated in various plans, tailored to the particular project.

Publish Plans

Currently one report and two types of plans can be generated based on the results of this methodology, a Development Plan and a Product Assurance Plan. The report specifies the results of the planning and estimation activities, providing COCOMO II values, Control level, documentation requirements, IV&V recommendation and a narrative describing the development activities to be performed based on the control level. Each of the plans utilize these results by tailoring a detailed plan for a critical control project and removing irrelevant requirements and information. These plans should be further tailored by the project manager to add information relevant to the project and organization, i.e, personnel, unique processes, etc. The majority of the

tailoring can be performed in the templates used to generate the Product Assurance Plan.

Once this is completed, the plans should be made available to pertinent people and organizations.

Development Plan

The development plan, based on MIL-STD-498 and Data Item Description DI-IPSC-81427, addresses each of the development phases and documentation to be generated by the project.

Product Assurance Plan and Level of IV&V, Independent Assessment or IV&V

The Product Assurance Plan addresses typical product assurance activities that should be performed during each of the phases and estimates the amount of time required to perform these tasks and evaluations. It also identifies the level of IV&V the project requires from independent assessment to full IV&V activities.

Conclusions

This approach combines all of the above to yield the outputs necessary to put together a successful project, namely:

- Identified software risks. The risk lists will take into consideration risks associated with software failures on previous NASA and aerospace missions (lessons learned, failure reports, defect profiles, etc.). This includes the identification of software components and intermediate deliverables by level of system criticality.
- An optimized plan that identifies software IV&V approach, development, and QA activities that mitigate and eliminate software risk for a given project at various times during the lifecycle.
- Consistent cost and schedule risk reduction budget estimates that establish a responsible balance between constrained project funding and the safe implementation of software subsystems.
- An equitably negotiated IV&V, Software Development, and QA plan that includes the priorities of the primary stakeholders while maintaining a high integrity program.
- IV&V, QA, and project plans that are compliant with institutional policies, ISO based Software Development Process Descriptions, and best practices from (for example) CMM.

The implementation of the tool support for the above process accommodates existing best practices for the various elements of project planning, estimation and assessment.

Acknowledgments

The research described in this paper was undertaken by Science Applications International Corporation (SAIC), and the Jet Propulsion Laboratory, California Institute of Technology, under contracts with the National Aeronautics and Space Administration, and by NASA Glenn Research Center. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, the Jet Propulsion Laboratory, California Institute of Technology, NASA Glenn Research Center, or SAIC.

The authors gratefully acknowledge the contributions, guidance and assistance of Steve Cornford, Robert Dugas, Marcus Fisher, Michael Greenfield, Frank Huy, Hoh In. Jim Kiper, Tim Larson, Kenneth McGill, Tim Menzies, Burton Sigal, and Siamak Yassini with special appreciation to John Kelly and Martha Wetherholt for their support of the research.

QWE2000 Vendor Technical Presentation VT8

Ruud Teunissen (Gitek)

TWeb: Testing e-Business Applications

Key Points

Key Points to be supplied.

Presentation Abstract

Abstract to be supplied.

About the Speaker

Since 1989 Ruud Teunissen is employed in the testing world. He has been involved in a large number of ICT projects and has performed several functions within the testing organisation: tester, test specialist, test advisor, test manager, etc. Based on his experience, Ruud participated in the development of the structured testing methodology TMap® and is co-author of several books on structured testing. The last years Ruud is involved in implementing structured testing within organisations on the Belgian and Dutch market. At this moment Ruud is working in Belgium for Gitek n.v. as Manager Testen. Ruud is frequently speaking in Benelux and Great Britain.

E-Testing

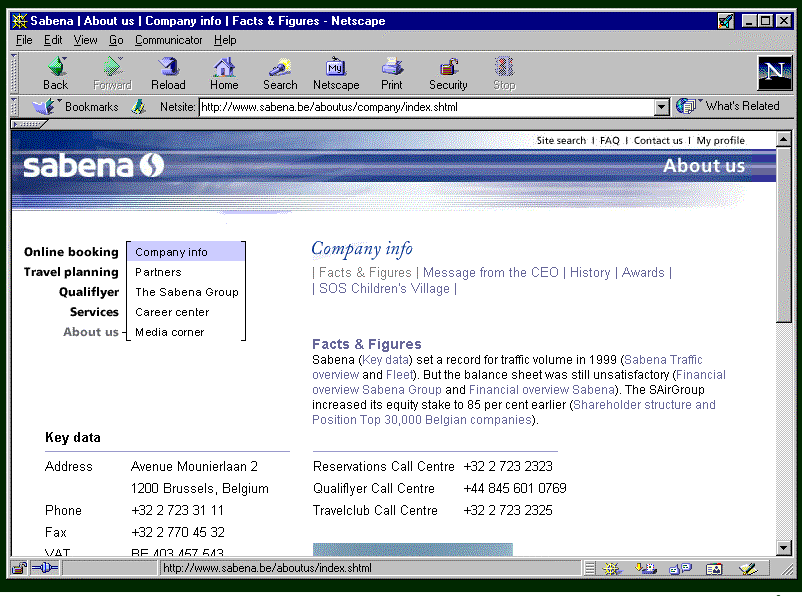
what e-xpertise do you need?

Ruud Teunissen
Gitek nv *interaction through software*
http://www.gitek.be
gitek@gitek.be

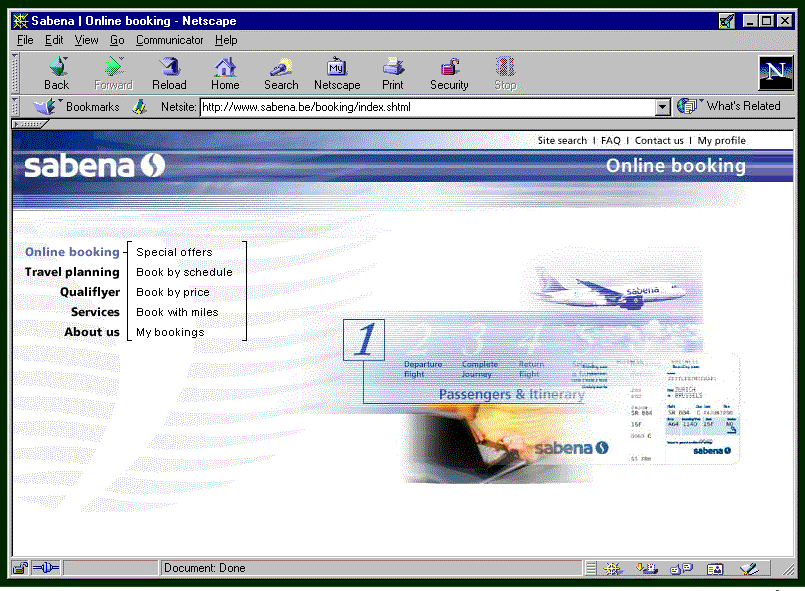
Agenda

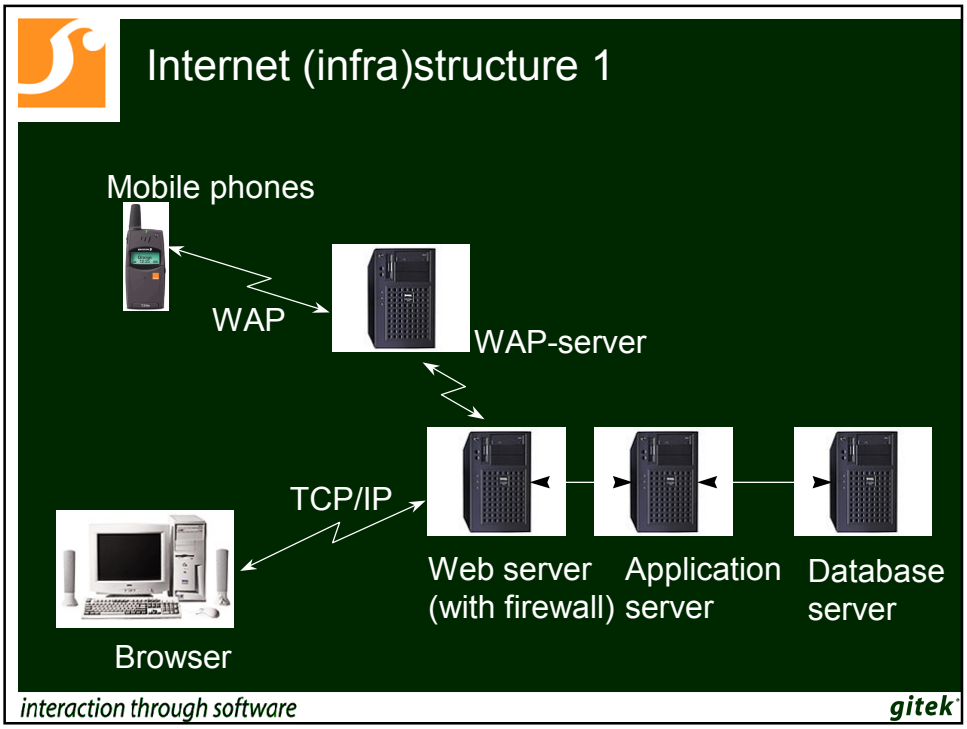
- Internet (r)evolution
- Why test e-business?
- How to test e-business?

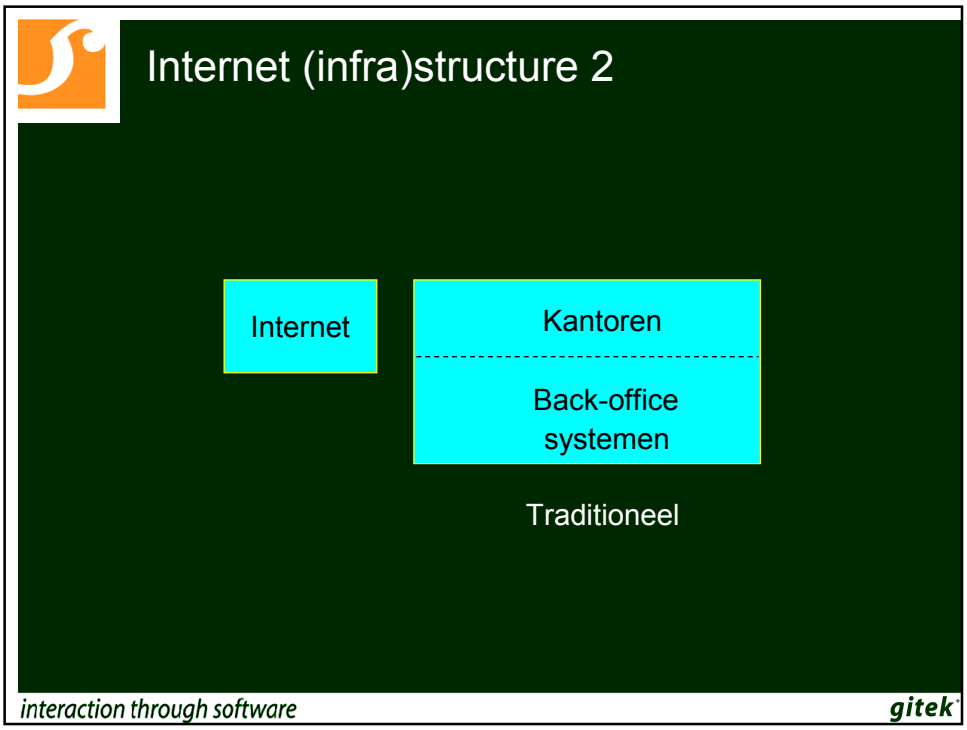
From simple functionality ...



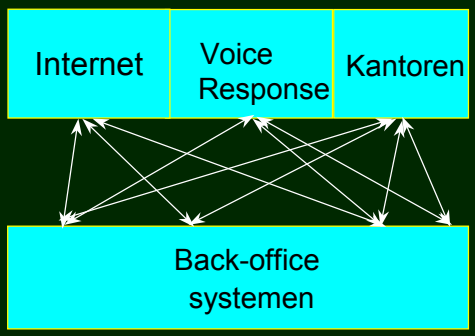
… to complex functionality



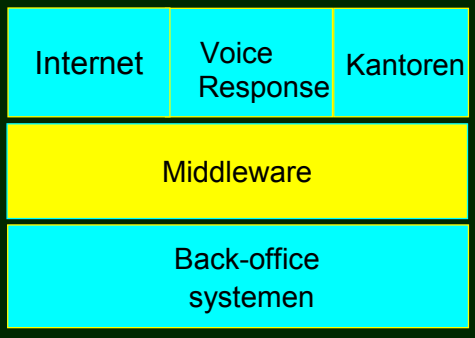

Internet (infra)structure 1
Mobile phones
WAP
WAP-server
TCP/IP
Web server
(with firewall)
Application
server
Database
server
Browser
interaction through software
gitek


Internet (infra)structure 2
Internet
Kantoren
Back-office
systemen
Traditioneel
interaction through software
gitek

Internet (infra)structure 2



Internet (infra)structure 2



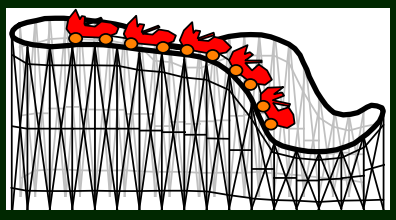
Multi Channel Architecture

Agenda

- Internet (r)evolution
- Why test e-business?
- How to test e-business?

E-business risks

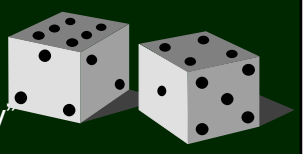
- Site is too slow or is down
- User unfriendly
- Hacker attack
- Infrastructural restrictions (browsers!)
- Insufficient organisation
- Faulty processing



Why test e-business?

- Historically: little testing
- Increasing risks
- Necessity of testing increases:
  Testing gives insight in quality of software
- Test strategy =
  testing versus risks (for every quality attribute)
- Risk = chance of failure * damage

Risk: chance of failure factors



- Ever increasing complexity
  - integration with back-office
  - more functionality
- Time-to-market, “delivery due yesterday”
- Insufficient knowledge and experience:
  - developers and testers
  - customer
  - specific aspects like security and usability
- Insufficient documentation
- New working method for organisation
- Unknown users, hardware, software
- Continuous change

Risk: damage factors

- Damage gets higher
  - more transactions
  - transaction is of higher value
  - missing "market share", because users
    - can't find site
    - don't do business
    - lose confidence and don't return

Agenda

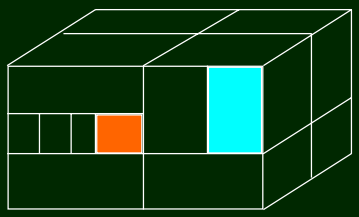
- Internet (r)evolution
- Why test e-business?
- How to test e-business?

Testing Strategy

Aim: To detect the most important defects as soon as possible at the lowest price!

Dependencies:

- Risks (necessity)
  - business
  - project
  - test
- Quality attributes
- Available resources



Test strategy

risks

↓

determine importance quality attributes

↓

determine importance subsystems

↓

determine test techniques

Quality attributes

- Which are (in general) of importance?

Dynamic
- security (functionality)
- accuracy (f)
- suitability (f)
- usability
- reliability
- time behaviour (efficiency)

resource utilisation (e)

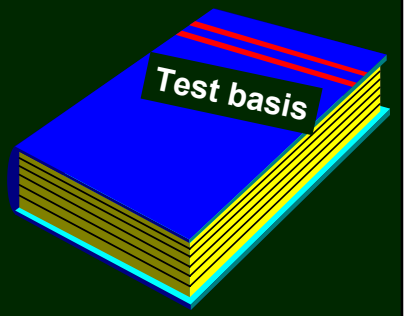
Static

interoperability (f)
- maintainability
- portability
- operability (usability)

ISO 9126

Test basis

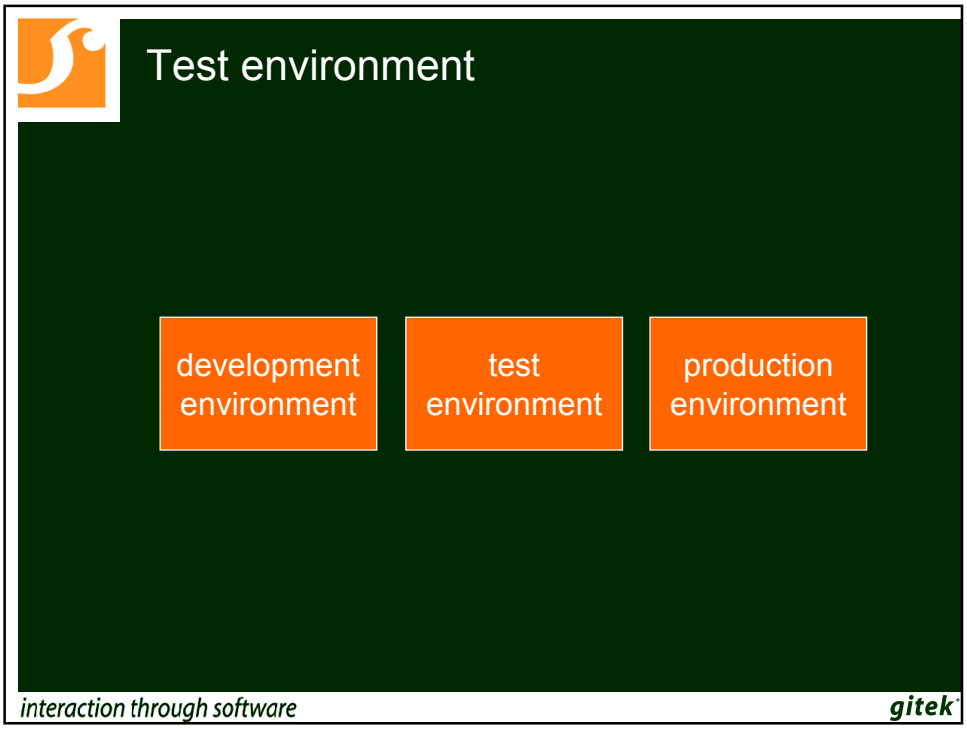
- Not readily available
- Kinds:
  - Requirements (business scenarios, use cases)
  - FD/TD (often very “light”)
  - Norms & standards
  - External references
- Much “digging” required, interviews
- Sometimes measuring i.s.o. testing

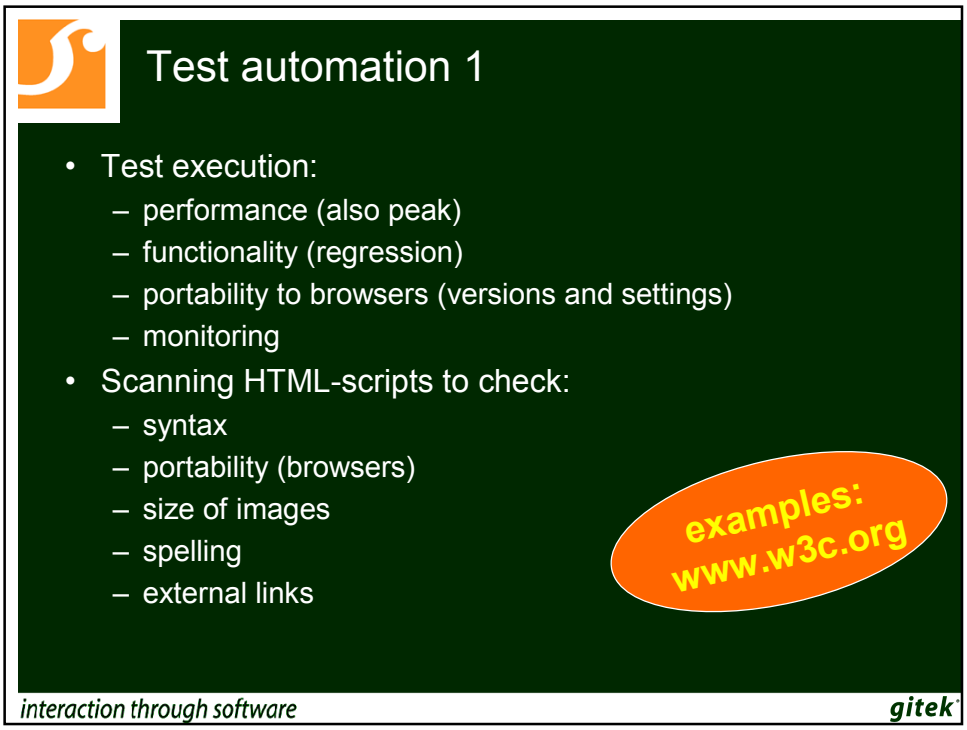


Test techniques

Many checklists

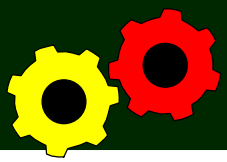
- Informal techniques: error guessing, DFT
- Performance test
- Security: audit, hacker software, penetration test
- Usability
  - lab with simulation
  - SUMI and WAMMI checklists


Test environment
development environment
test environment
production environment
interaction through software
gitek


Test automation 1
• Test execution:
– performance (also peak)
– functionality (regression)
– portability to browsers (versions and settings)
– monitoring
• Scanning HTML-scripts to check:
– syntax
– portability (browsers)
– size of images
– spelling
– external links
examples:
www.w3c.org
interaction through software
gitek

Test automation 2

- Test tools
  - Load & Stress
  - Internet specific C&Pb, big brands + new brands
  - challenges:
    - moving pictures with applets, and so on
    - number of changes in pages
  - HTML-check tools (some free, some cheap via the Web)



Knowledge and skills

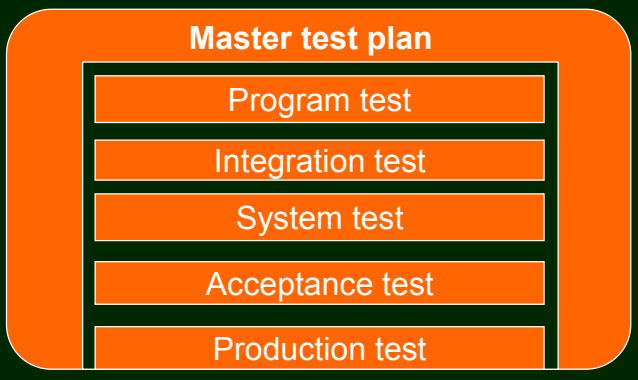
- Testing
- Business: specs often missing!
- Internet, system and architecture: what is standard?
- Test tools: Capture&Playback + Load&Stress + HTML-tool
- Specific (for instance, security and performance)

And we keep on testing … the production test

- Periodically testing:
- Performance monitoring
- Integrity check
- Security monitoring
- and so on

TESTING as control activity

Test levels in master test plan



NEW: production test during exploitation phase

Test trends

- More mature development process
  - More requirements and specifications
  - More test environments
  - Configuration management
- TMap keeps helping...
  - Life-cycle model with testing strategy
  - Techniques
  - Infrastructure with tools
  - Organisation

Conclusion

- E-business testing requires its own test approach
  - Focus on testing non-functional quality attributes
  - Test automation very important
  - Black-box testing and white-box testing mix
- Specific kinds of e-xpertise necessary

THANK YOU!

QWE2000 Session 9T

Dr. Rainer Stetter [Germany]
(Software Factory & ITQ GmbH)
http: www.itq.de, email: stetter@sf.com

"Test Strategies for Embedded Systems (9T)"

Key Points

- Real life example (control system), project running from 1997 to 2000
- Detailled discussion of every project phase
- Recommendation of approaches, measures and tools

Presentation Abstract

Embedded Systems are used more and more in the field of mechanical engineering. This situation leads to an increasing demand for efficient test strategies for Embedded Systems. In my presentation I would like to demonstrate our approach with a real life example. In the example IËll discuss the test strategy for a PC104 based control system for a material testing machine.

About the Speaker

At the Technical University, Munich I studied mechanical engineering and as I was interested in software engineering I took some classes in computer sciences. While I was doing my PhD in developing a robot simulation system, which I got in 1993, I improved my knowledge in software engineering. From 1993 until 1997 I was working as a Research & Development Manager at Zwick Company, Ulm - Germany.

Since 1997 I have been one of the General Managers of Software Factory GmbH, Munich. In addition, since 1998 I have been working with the Munich based firm itq GmbH as a General Manager. Together, Software Factory GmbH and itq GmbH form the Software Quality Center. With some partners of the Technical University of Munich and VDMA (German Machinery and Plant Manufacturers' Association), we work on approaches to improve the quality especially in the field of embedded systems.

Since 1998 I'm Vice President of the VDMA Software department.

ITQ Information: Technology: Quality

SF Software Factory

Test Strategies for Embedded Systems

Dr.-Ing. Rainer Stetter
www.itq.de

Example: Material Testing

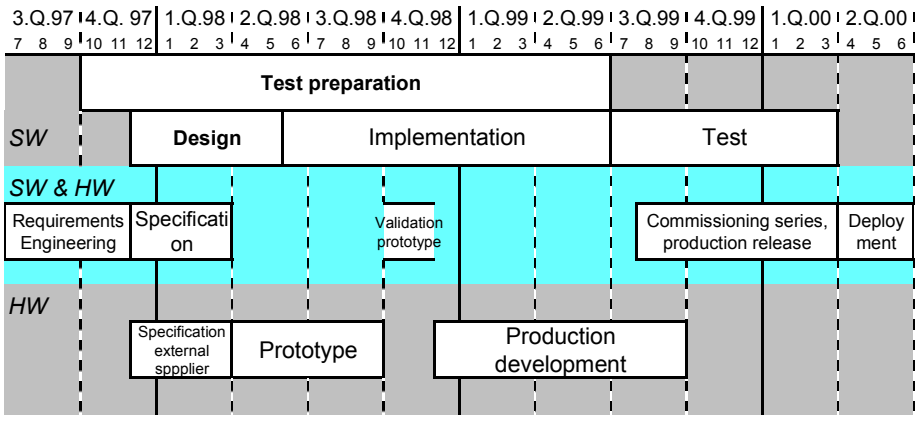
Quality Requirements

- Safety critical
  - Medical devices
  - Life and health
- Quality Requirements according to the FDA
  - stable and detailed process
  - Validation plan

Phases of the Validation Plan

- Splitt whole process into different phases
  - Requirements Engineering
  - Specification/Design
  - Implementation
  - Test
    - Module
    - Integration
    - Acceptance
    - System
  - Deployment

Time schedule (first version)



Extract of a validation plan

	Validation activity	Rating				Reference	Responsible
		1	2	3	4		
1.	Definition of the user functions of the software		x			..\..\..\Analyse\review\A1-Funktionen der SW.doc	uss-kr
2.	Definition of the required input data		x			..\..\..\Analyse\review\A2-Eingaben der SW.doc	uss-kr
3.	Definition of the required output data			x		..\..\..\Analyse\review\A3-Ausgaben der SW.doc	uss-kr
4.	Definition of limits, defaults, special input which have to be accepted by the software		x			..\..\..\Analyse\review\A4-Bereiche für Ein-Ausgaben.doc	uss-kr
5.	Definition of the required performance		x			..\..\..\Analyse\review\A5-Performance der SW.doc	uss-sj
6.	Definition of the external interfaces and the user interface		x			..\..\..\Analyse\review\A6-Schnittstellen der SW.doc	uss-hib
7.	Definition of error classes		x			..\..\..\Analyse\review\A7-Definition von Fehlerklassen.doc	uss-kr
8.	Definition of the reactions on error classes		x			..\..\..\Analyse\review\A8-Reaktion auf Fehler.doc	uss-kr
9.	Definition of the system environment		x			..\..\..\Analyse\review\A9-Betriebsumgebung.doc	uss-sj
10.	Definition of safety requirements, functions and features	x				..\..\..\Analyse\review\A10-Sicherheit der SW.doc	uss-bi
11.	Software hazard analysis		x			..\..\..\Analyse\review\A11-SW-Gefahrenanalyse.doc	uss-bi
12.	Traceability of system and software requirements		x			..\..\..\Analyse\review\A15-Quercheck-System-SW.doc	uss-sj
13.	Design of the system test plan			x		..\..\..\Analyse\review\A17-Systemtestplan.doc	sf-rs
14.	Design of the acceptance test plan			x		..\..\..\Analyse\review\A18-Abnahmetestplan.doc	sf-rs

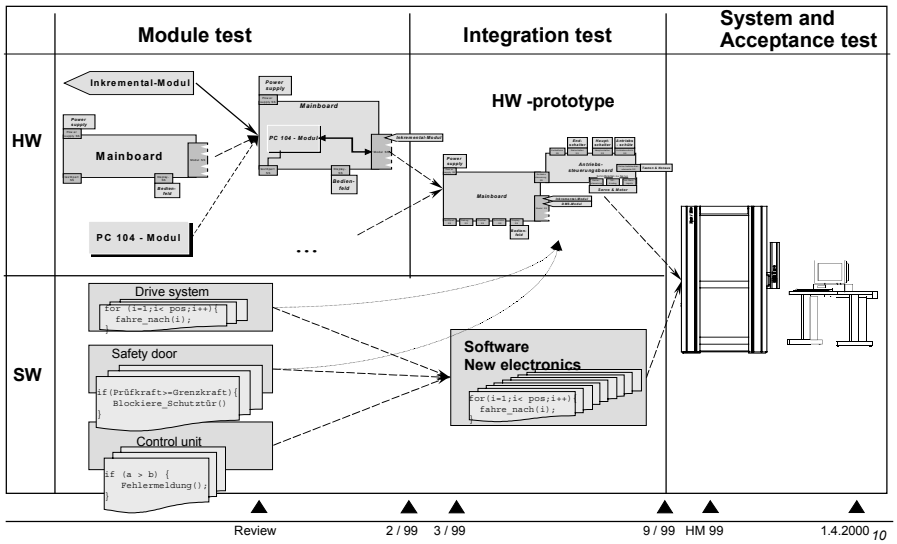
Requirements Engineering

- Description of Requirements
  - Prose – Non-formal (50 pages)
    - Formalisation subsequently, but not detailled enough
  - both forms are sensible
    - Prose – Management
    - Formalized – Test
- Hazard Analysis
  - of the system with special focus on software
- Test plans
  - Almost impossible to do in a sensible way
- Trial and error group

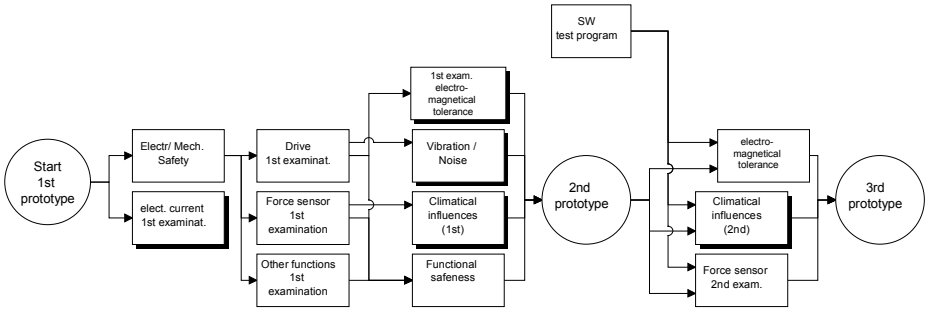
Specification and Design

- Specification
  - One modularized document (500 pages) in prose
    - No additional work for hardware group
  - Written from developers for developers
    - Inconsistent level
    - Not formal enough
    - Hard to develop test procedures
- Test
  - Structure and elements of tests were designed
  - Planning of the integration test
- Software design
  - Guideline for software design
  - UML with weekly design reviews

Structure of the test activities



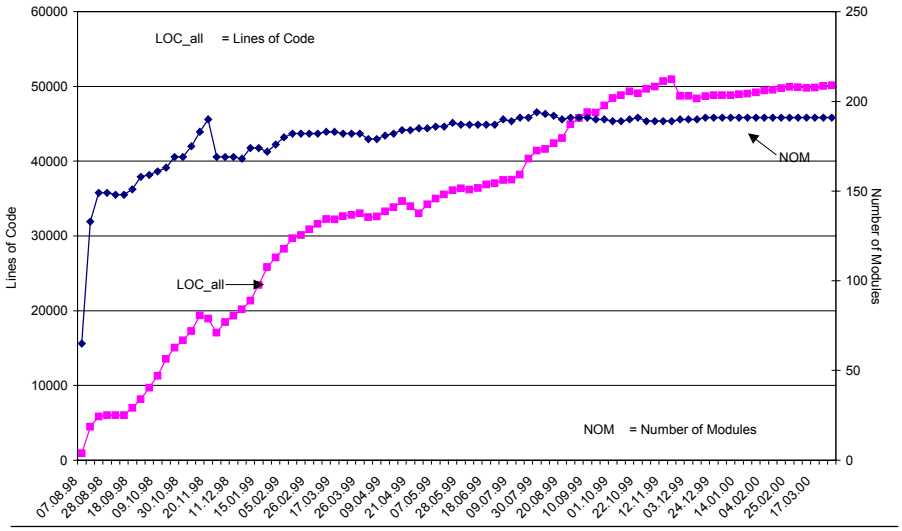
Structure of integration test plan



Implementation

- Organisation
  - Daily check in and build
- Weekly code reviews
  - Social aspect
    - Code is no longer a personal secret
    - Same philosophy / style of coding
  - Technological aspect
    - Only low level errors
      - Reading was too fast -> Tom Gilb
- Test
  - Development of test procedures and approaches

Number of Modules and Lines of Code vs. Time

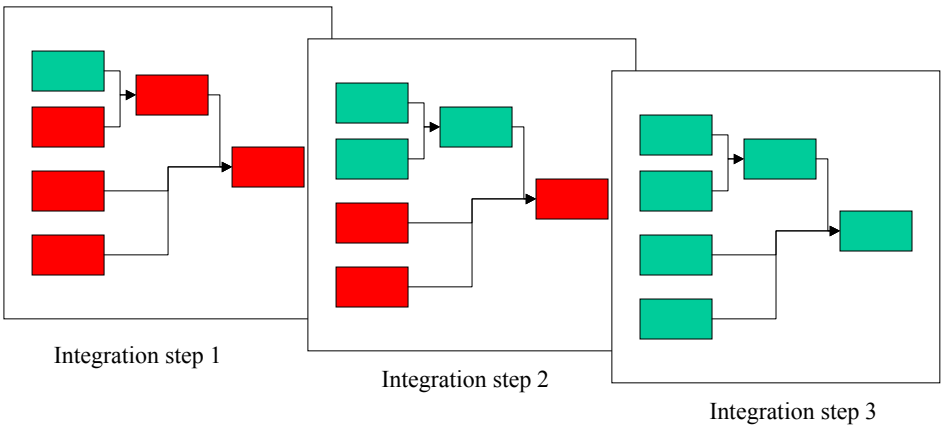


ITQ Information: Technology: Quality
SF Software Factory
Test Strategies for Embedded Systems

Integration test

- Theory and practice in planning
  - Delays of software or hardware
    - Very hard time
    - A lot of improvisation
- Design of test plans
  - Have to be modular -> flexibility to adapt to new situation
- Control
  - Regular tracking is very important
  - Visible and understable even for the management
- Code Coverage (starting with module testing)
  - Whole system not testable at once
  - Had to be divided into parts

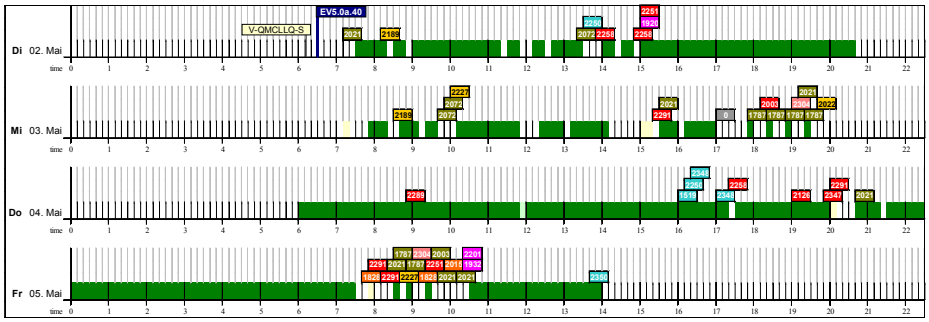
Tracking of integration test



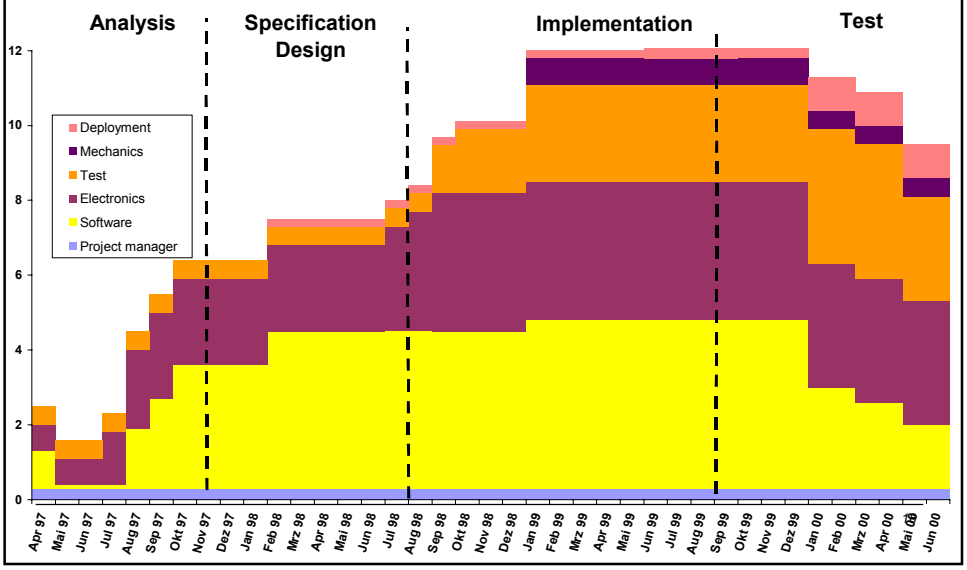
System and acceptance test

- Test environment
  - Most critical aspect
    - New hardware was not available in required numbers
    - Old machines had to be modified
  - System and acceptance test ran concurrent
- System reliability and defect tracking
  - Very detailled tracking
  - Wide ranging set of tests on technical aspects
    - Humidity
    - Lack of electronic current
    - Different temperatures
    - Safety

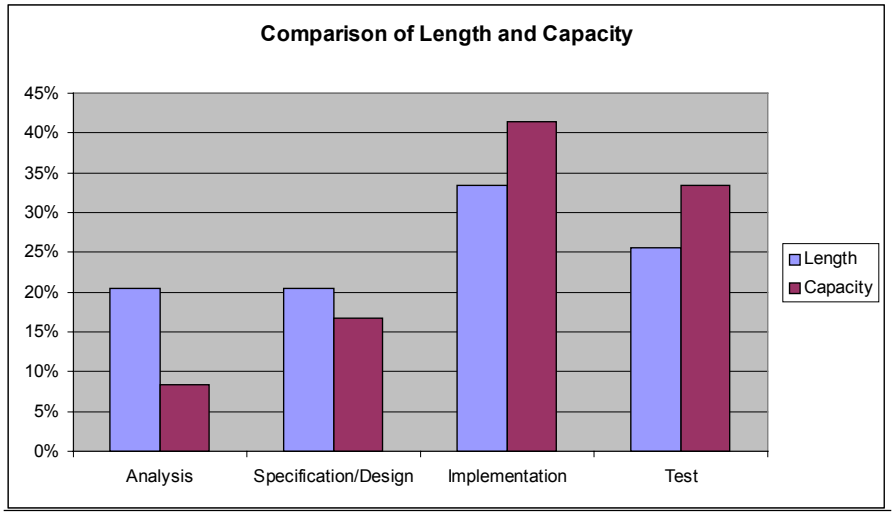
Tracking of the tests



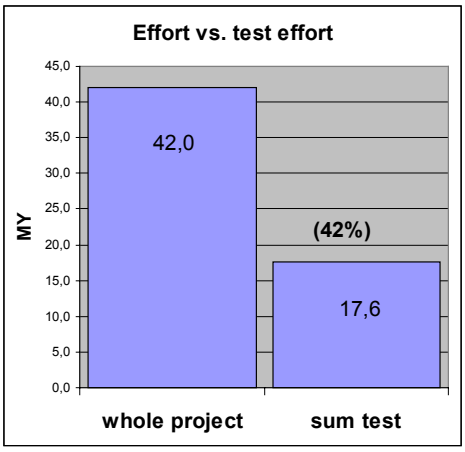
Overview of efforts

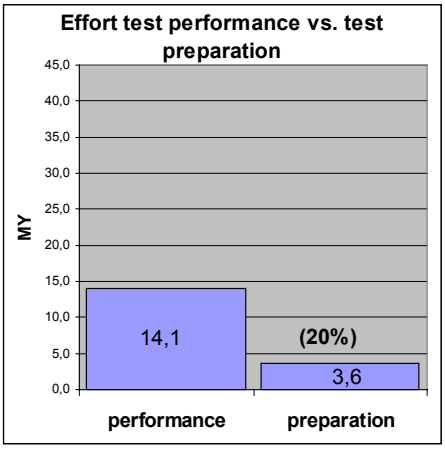


Length of phases vs. manpower in phases



Test effort





Lessons learned (I)

- Validation plan
  - gave structure for the whole project
    - Most useful for the early phases
      - Responsibility for disliked activities
    - Some bureaucratic overhead
  - Reviews very helpful
    - Phases
    - Design and coding
- Design
  - UML is sensible
  - No completion of design without coding

Lessons learned (II)

- Test
  - To start test planning early is very hard, but sensible
    - Modular design of test/test procedure is very important
      - Re-useability of test patterns in critical situations
  - Tracking through all phases -> management
    - Test procedures
    - Defects and realibility
  - Environment
    - You never have enough

Conclusion

- Embedded Systems <-> inter-disciplinary teams
  - Need to encourage communication
    - Modules won‘t understand each other,
    - if people don‘t talk with each other
- To work in a structured and planned way
  - Doesn‘t prevent all problems
  - But makes it easier to handle them
- Think before you work !
  - Life will be easier

"Test Strategies for Embedded Systems"

Dr.-Ing. Rainer Stetter
ITQ – IT & Quality GmbH, Munich
http://www.itq.de

Introduction:

Embedded Systems are used more and more in the field of mechanical engineering. This situation has lead to an increasing demand for efficient test strategies for Embedded Systems. On the basis of a real life example I want to discuss our approach for testing embedded systems.

Example: Control System for Material Testing Machines

Material testing machines are used in a broad area. Any material has to be tested once before it's used in any construction. Therefore there is a high safety and security demand for the control system of a material testing machine. The control system which is to be discussed has to collect all machine data in real time which are used for the determination of the actual control parameters. The real time operating system used is VxWorks. The software runs on a standardized PC module with a 486 processor. The processor has to supervise ten input channels at the same time which are refreshed every two ms. For the design of the software we used UML, based on Rose98 including the code generation and the round trip engineering component.
The development process is to be validated by the FDA because this kind of machine is used for testing medical devices, too. According to the FDA demands we used a validation plan to manage all project activities. The validation plan describes all activities which have to be fulfilled in a project, including all phases from the requirements engineering up to the point of the product release.

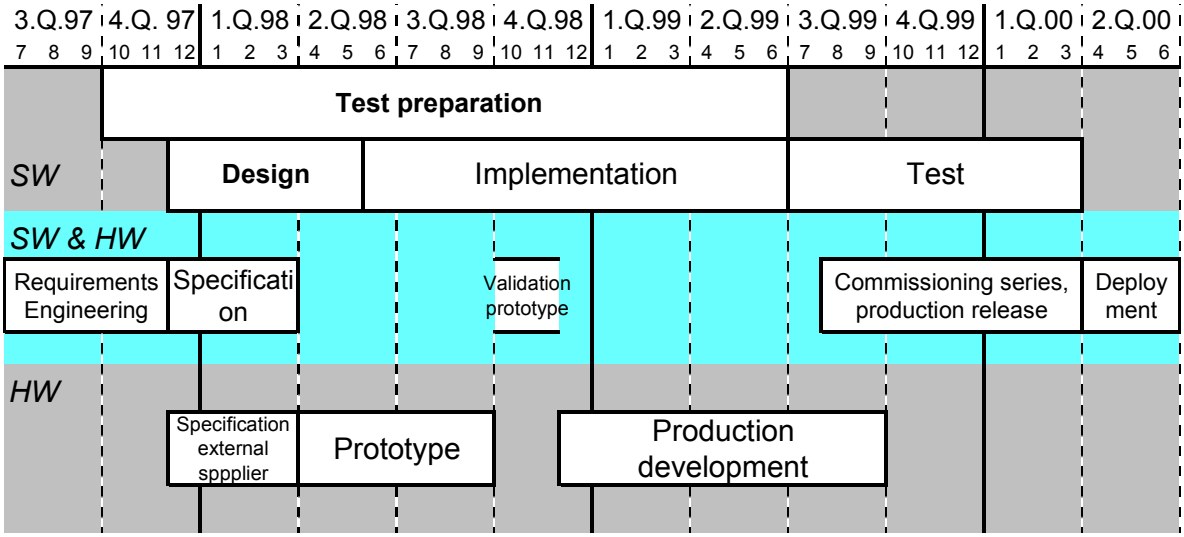


Fig. 1: Overview of the project schedule

Fig. 1 gives an overview of the project. The project was splitt into several phases. The preparation of the test started quite early and ran parallel to all other activities until the start of test phase.

According to the validation plan we had to work on a software hazard analysis in the **requirements engineering phase** to show how the system would react in the case of a severe software fault. This analysis influenced the system design in general. Because we couldn't guarantee an error free software system we had to provide a (mechanical) hardware backup component for any potential safety critical situation. Otherwise we would have had to use a two-processor system which was out of the question because of the additional costs. Another validation activity in this phase was planning the structure of the system and the final acceptance test. Even though you can read about software testing in almost any publication, that consideration of the test process should begin at a very early point in the project, we experienced that it is very hard to this in a effective way.

The description of the requirements was written in prose in a non-formal style. After a review of the requirements specification through the test group we formalized them subsequently. In the test phase we had to go through the experience that the level of formalization used wasn't detailed enough because the requirements were hard to test. To improve the testability of the requirements we will strengthen formalization. But you still need some requirements written in prose because it's hard to convince management or marketing people to read heavily formalized requirements.

After a review of the requirements engineering phase, in which all validation plan activities were discussed in detail, the design phase was started.

In the **specification and design phase** we had to determine in more detail which system function had to be implemented in hardware (electronics) or in software. In this phase it was quite difficult to keep the hardware and software team in touch. Both of these groups were ready to go into details even though their main job at that time was to determine the interfaces.

The specification document was written in prose, too. It contained about 500 pages. To admit concurrent working of several persons we organized the document to be modular. To write a specification document was neither new nor additional work for the hardware group. They were used to work in this way. To convince the software team not to start with more exciting things was quite hard. In the test phase we had to repeat the experience that our work could have been better. Once more we had found out that the specification should be more formal to make it easier to develop good test procedures and test cases, in addition we had to recognize that the level of abstraction was inconsistent.

For the software design we introduced UML. As a matter of fact, the use of the Rose98 was very helpful. The sequence diagrams were especially useful because in this view the dynamic behaviour of the system was quite easy to model. After a period of familiarization, the software engineers worked in small groups interactively with the tool. This was very efficient because the communication between the software engineers was highly

stimulated. In addition the software design was reviewed weekly. The design reviews were continuously attended by a member of the test team.

In this phase the test team worked on the design of all test activities. The structure and the interdependencies of the test activities are shown in fig. 2.

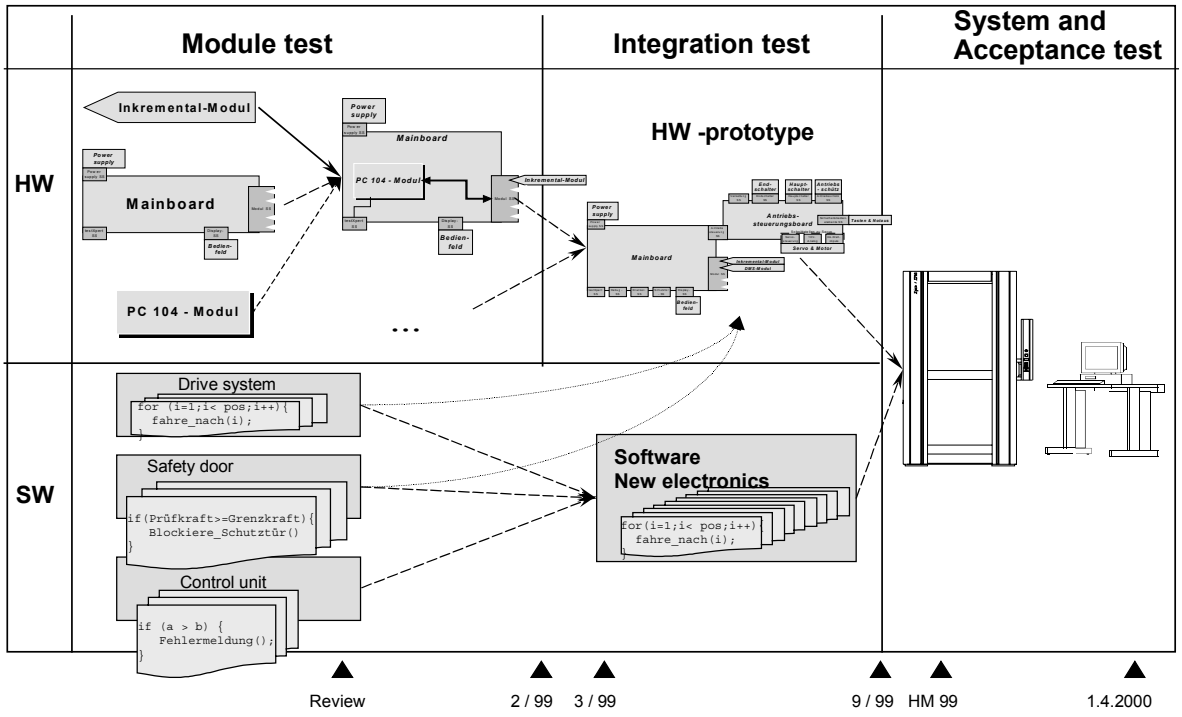


Fig. 2: Structure of the test activities

To describe a module test for hardware modules was quite easy. Due the OO-design of the software it was almost impossible to design a module test for the software. Therefore we decided to run code reviews and to do code coverage tests through the implementation phase to reduce risks and improve software quality. According to the module test plan an initial version of an integration test plan was developed. For the visualization of the interdependencies of the modules and the different test procedures we used a flow chart graphic, see fig. 3.

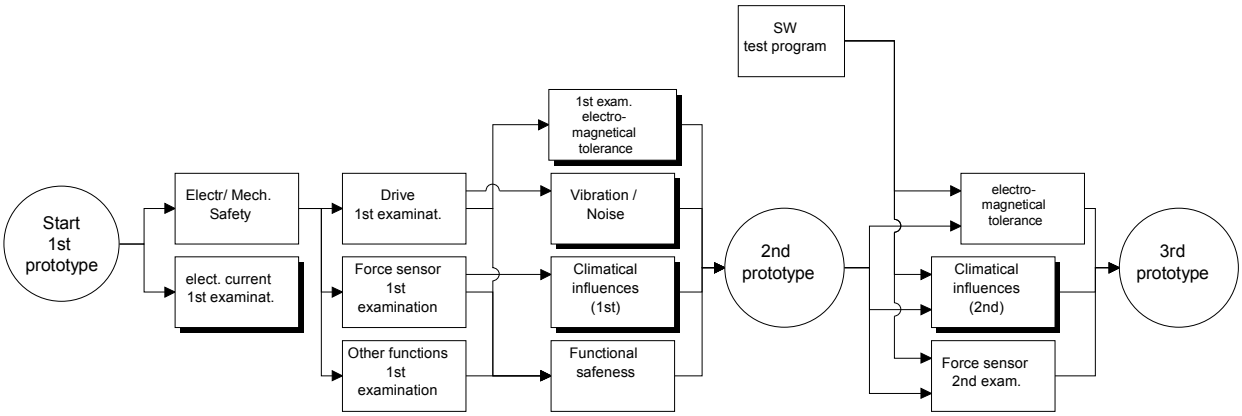


Figure 3: Structure of the integration test plan

During the **implementation phase** the different levels of the test plans were detailed. The weekly code reviews were attended by a member of the test team, too. Together with the software engineers the test team worked on the preparation of the white box tests. For the calculation of the code coverage we used CodeView. At the beginning of the project we planned an almost 100% coverage. After a while we figured out that it was more helpful to concentrate on some critical modules than to try to do all at once.

At the beginning of the implementation phase we thought that we already designed around 90 % of all needed classes (modules). But we had to figure out in the first weeks of the implementation that there was still a lot of detail work to be done. Therefore there is a steep increase of the number of classes in fig. 4 at the very beginning. After some weeks there was a stabilization of the number of modules. On the other hand there was a more or less steadily increasing number of lines of code.

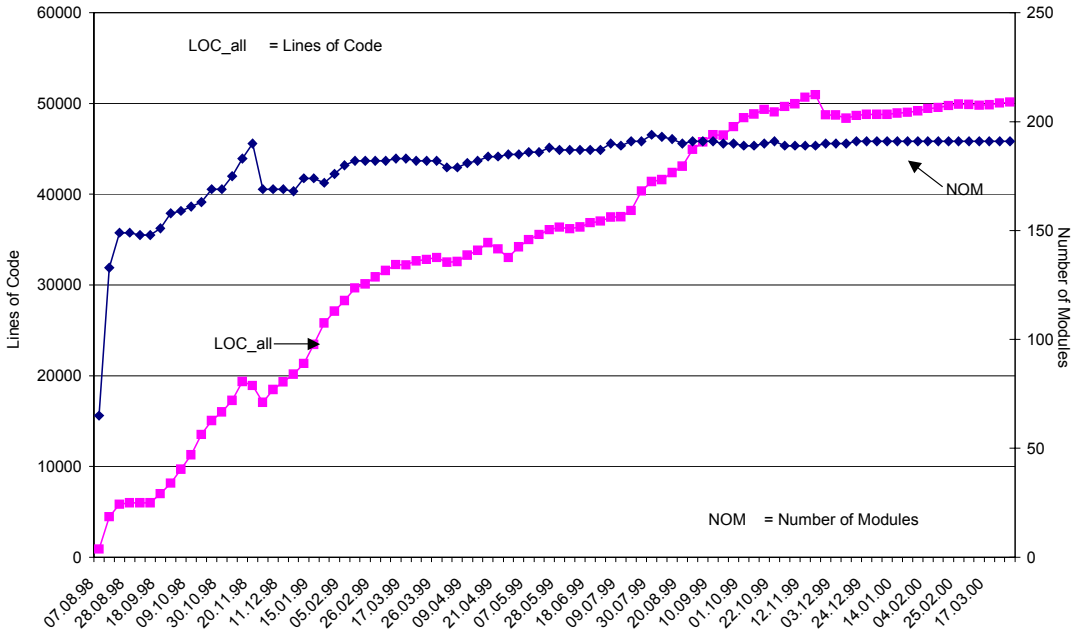


Fig.4: Number of modules (classes) and lines of code vs. time

The **test phase** started with the testing of some hardware modules. This period was quite tough because on one day the situation occurred that a hardware module was completed but the corresponding software wasn't finished. On the next day for another module the software was finished but the hardware was delayed. Looking back over this period we had to learn to accept that even with the best planning some improvisation can't be avoided. To be flexible in this situation you have to design your test procedures quite modularly. It has been our experience that the templates which are suggested by the IEEE were a good basis for tailoring our own templates.

After getting to a certain level of maturity in the basic modules we started with the integration of the system. To track the progress of the integration we used a coloured flow chart graphic.

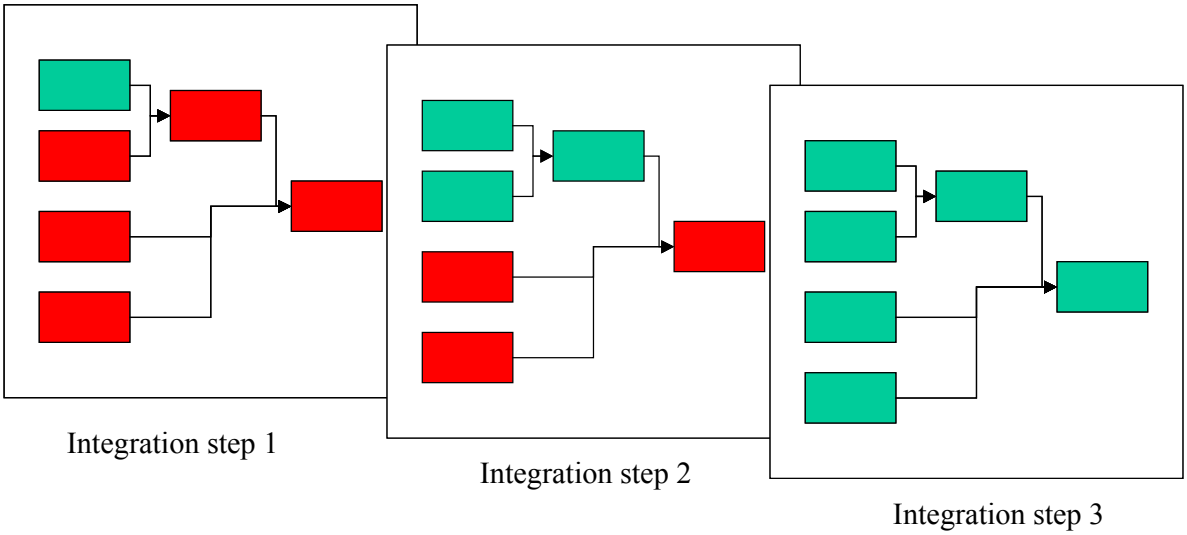


Fig.5: Progress tracking of the integration
(released modules are shown in green, unreleased modules are shown in red)

The basic idea of our approach is shown in fig. 5. For every module which is represented by a rectangle in the plan we tracked general information, such us the version, the date of release, the name of the person who released the module and the test procedures which were the basis for the release of the module. In addition we defined in advance which modules had to be tested in each integration step. In matching every integration step there were some integration test procedures which had to be performed. Through this visualization we always had a good overview of the actual situation. This was especially helpful in discussions with the management. After getting to the final point of the integration test we started with the system test. To do the system test we used a modified machine from a previous generation. After some weeks we got a pre-release of the new hardware module (mechanical and electronic components) so we could start the acceptance test concurrently with the system test.

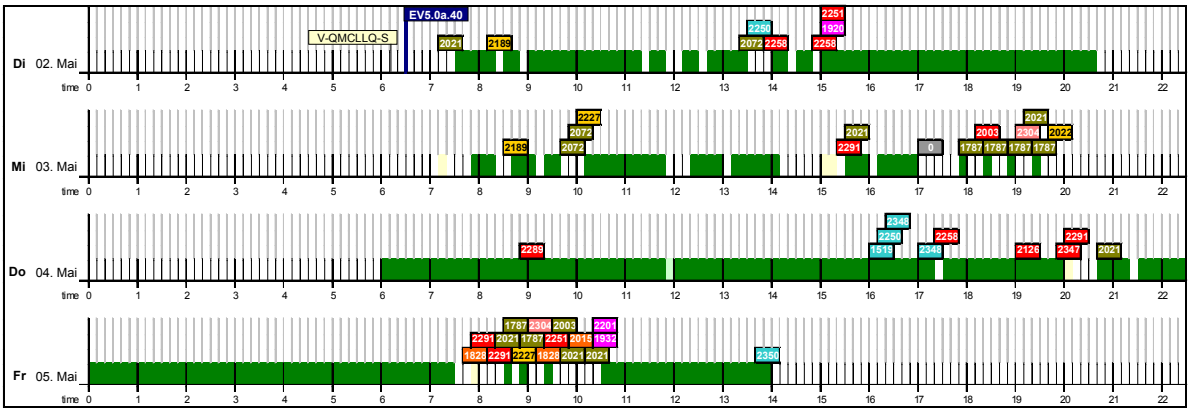


Fig 6: Tracking of the system performance and the occurrence of defects

The system performance and the defect rate were tracked by a spreadsheet with the format shown in fig. 6. The green colour shows that the system ran without any problems. All defects are marked by a rectangle. The colour of the rectangle identifies the severity of the defect and the most likely contaminated module. The number in the rectangle refers to the defect identifier in the error database.

Based on this type of spreadsheet and some other interpretations of the test results the test team could come to the decision whether or not the product was ready to release.

Effort:

The whole project was designed to learn how to manage projects of this type. One of the key questions of project management is, which effort has to be invested in which phase and of which team. According to this goal we measured not only technical aspects but also some management aspects. Figure 6 shows the progression of the ongoing project. The effort of the project staff was measured in percentage of theoretically available man power. We found out that a realistic percentage of real working on the project is for developers about 60-70%.

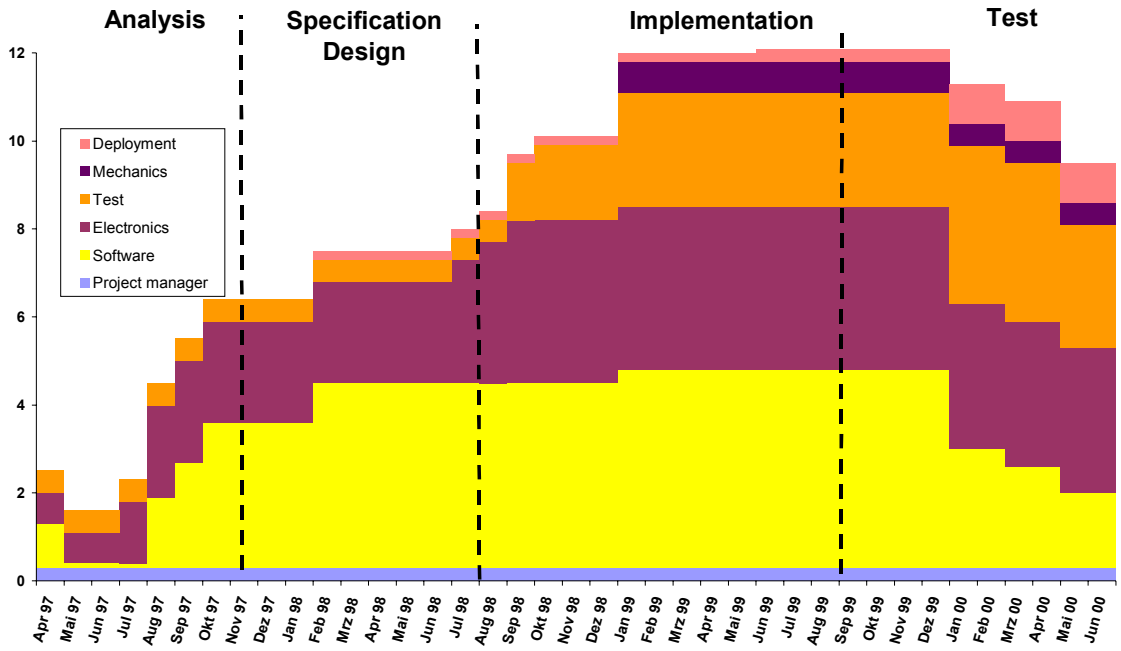


Figure 6: Overview of the efforts

The shape of the curves indicates that the product release worked quite well because there is a decline of the effort at the end of the project. Projects which end in a crisis have normally a steep increase in the effort by about two thirds of the theoretically estimated end date.

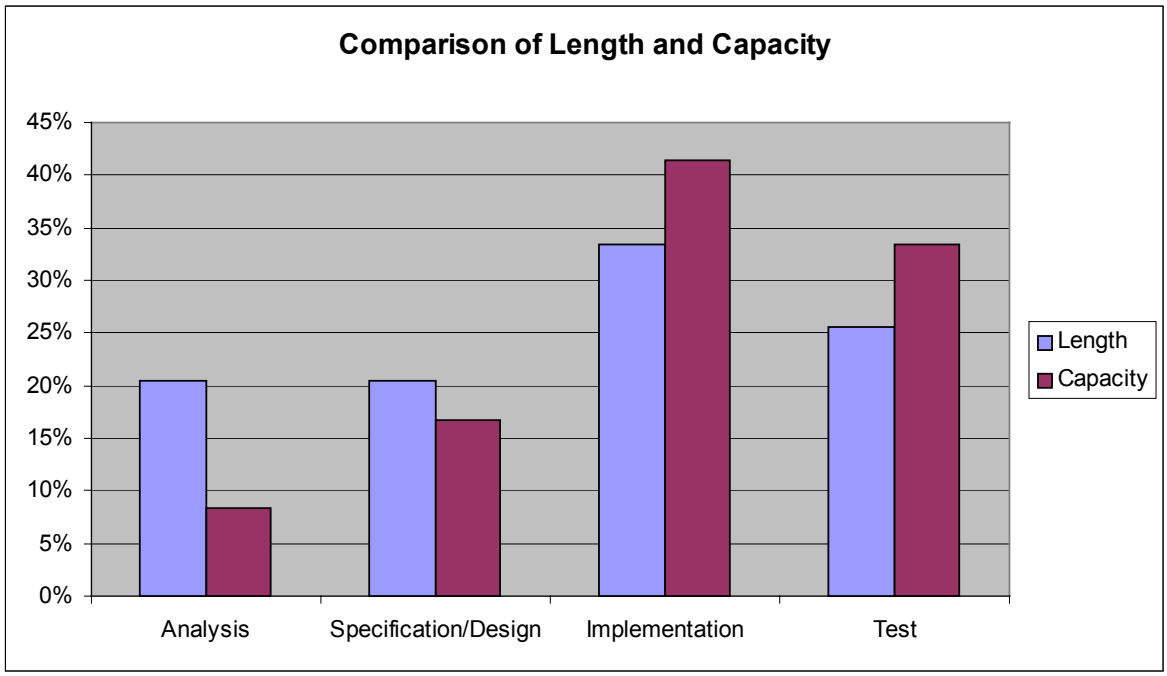


Figure 7: Length of phases vs. manpower in phases

Fig. 7 compares the lengths of the periods of time with the manpower invested in the different phases. The first phases seem to be quite long, but a look at the percentage of the manpower spent shows that the first phases needed only a small portion of the whole effort. We spent about 25% of the overall effort for requirement engineering and design but needed about 40% of the time. We think this ratio is quite realistic, because there is a big demand for interdisciplinary discussion in the first phases. These discussions are the basis for decisions and important decisions normally need time.

Finally we compared the test effort with the overall effort. In addition we figured out how much we invested in test preparation and test performance. The results are shown in fig. 8. The overall effort was about 42 man years. In test activities we spent about 42%. 20% of the test effort was needed for preparation, the rest to perform the tests.

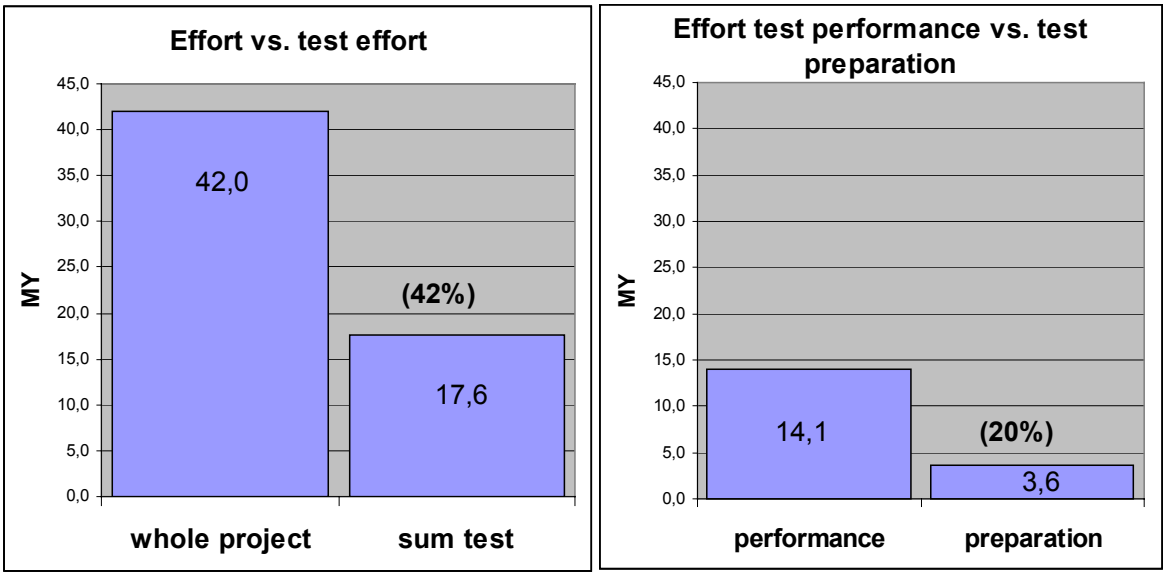


Figure 8: Test effort

Summary:

The project described ran from July 1997 to completion some months ago with a delay of only about 4 weeks. This result underlines the fact that collateral test activities conducted through the whole project are very helpful. But on the other hand it demonstrated that it may be quite hard to do meaningful test planning in a very early phase of the project. It's fairly predictable that the basic test approach might be quite wrong. Therefore it's very important that the structure of the test documents is modular to give the flexibility of adapting your test procedures very quickly to real circumstances.

Finally, we can conclude that the strategy of using a validation plan was very sensible. Firstly it gave the whole project a certain framework and some regulations. Secondly it forced us to start our test activities in a very early phase of the project which was eventually one of the preconditions for project success.

We spent a long period of time and a lot of money to prepare and plan the project in a sensible and serious way. In the eyes of the management this philosophy takes too much time for things which you can't see. Therefore it's very important to inform the management regularly and in a way which is understandable for non-insiders. Otherwise you are quite soon forced to work on getting understandable results rather than to spend time for thinking. Even though everybody knows that life is easier if think before you act.

QWE2000 Session 9A

Dasha Klyachko [UK]
(Allied Testing)

"Specifics of E-Testing: Difference Between Traditional and On-Line Software Development and Its Effect on Testing (9A)"

Key Points

- Software testing
- Changes in the testing process
- Software life cycle

Presentation Abstract

This paper first reviews the difference between the software development process for the traditional and on-line applications found through the series of interviews with web developers and development managers. It then suggests the consequent changes in the testing process and outlines several new features of e-testing.

The theoretical framework is based on the analysis of the system life-cycle (SLC) (Yourdon, 1989, Clarke, 1997) and its modifications influenced by the changing business needs, effects of globalisation and standardisation of underlying technologies. The consequent conclusions about changes in the testing process is based on the author's work experience and interactions with the colleagues in the industry.

The paper might be of interest to the practitioners planning and executing testing of on-line applications.

About the Speaker

Dasha Klyachko

Dasha is doing her PhD at the London School of Economics. Her academic research concentrats on Outsourcing in the IT industry (E-commerce in particular). Research interests include web certification methodologies and specifics of the development and testing for on-line applications.

Dasha also is a founder and a managing director of Allied Testing - an outsourcing company specialising in on-line testing of Internet-based applications. Prior to founding Allied Testing, Dasha spent most of her career in California as a software testers/developor and system analyst.

Dasha lives in London, married, has a daughter.

Specifics of e-testing: difference between traditional and on-line software development and its effect on testing

Dasha Klyachko
Director of Allied Testing, LLC
PhD researchers at the London School of Economics,

24 Redcliffe Mews, London SW10 9JU
+44 (0) 7887 647 813
dasha@alliedtesting.com, d.klyachko@lse.ac.uk

Abstract
This paper aims to identify differences between traditional information systems and e-commerce systems from the point of view of software testing. The paper presents taxonomy of these differences and describes consequent changes in testing techniques. I have tried to create a four-dimensional framework of changes and make these dimensions as "orthogonal" as it is possible on the relatively limited set of data for the industry.

This article draws on my practical experience and on series of formal interviews with web developers and development managers. It is a survey article with which I am opening the discussion, defining the common ground for us to continue searching for a model defining place of a testing industry inside the vast area of information systems.

Keywords: software testing, web testing, testing process, software life cycle.

1 Introduction

I will define *e-testing* as 'testing of e-commerce applications' where *e-commerce* means 'conducting business on-line by means of Internet enabled communications'.

Testing has always been an important area of the software development process. Changes in the technological and business environment and development of e-commerce applications require new testing methodologies. This paper aims to identify differences between traditional information systems and e-commerce systems from the point of view of software testing. It presents taxonomy of these differences and describes consequent changes in testing techniques.

In part 2 of this paper I identify four differences between traditional information systems and e-commerce systems from the point of view of software testing. In part 3 I describe consequent changes in testing process. In part 4 I discuss new trends and the future changes in e-testing.

2 Categories of changes between traditional testing and testing of on-line applications

The paper considers four types of changes:

1. Software life cycle and project structure changes (including outsourcing of development and testing work)
2. Externalisation of internal systems and sequential changes in operating requirements
3. Changing role of web systems - the iceberg effect
4. Media and system complexity.

2.1 Software life cycle and project structure changes

The experience of four decades of commercial software projects has resulted in a conventional approach to the management of application development, namely the system life-cycle (SLC) [1], [6]. Conventionally, SLC is divided into the following phases: project planning, requirements analysis, system design, construction, implementation, operation. There are various interpretations of the SLC. Some project management methods curtail the requirements analysis phase (these are typically based on IEEE standards and have a heavy 'software engineering' flavour). It is also common to release an application first to a small set of users in order to identify problem missed during the construction and implementation stages. Testing

conventionally followed the SLC through all the phases. Testers participated in the reviews of requirements and design, in the unit and system testing.

The competitive advantage is often huge for the first-comers. Therefore, e-commerce companies rush to the market at the expense of quality. Aggressive development deadlines often result in skipped requirements analysis stages. Prototypes are often released as production systems. Thus, testing is left out of initial stages of SLC. To offset lack of requirements analysis companies often perform usability assessments allowing users to contribute to the product requirements lists. Testing shifts to the later stages – to user acceptance and system testing. Moreover, even this type of testing is often squeezed to a minimum.

Externalisation of systems makes pilot projects rare and difficult to implement. Therefore, increasing importance of the usability testing. (I considered this change as part of the SLC changes because pilot projects are part of SLC. Another place for it would be under the system externalisation changes).

2.1.1 Outsourcing of development

Outsourcing of development often leads to incomplete testing coverage. There are two testing arrangements possible when development is done by a 3rd party. According to one arrangement the 3rd party shares testing responsibilities with the client's internal testing department. In this case, an internal testing department is responsible for integration testing and the 3rd party – for a functional testing of a new component. Co-ordination challenges between internal and external testing groups result in incomplete test coverage. They also decrease involvement of testers at different stages of SLC. For example, a development agency is not involved in the requirements analysis and system integration stages, while the internal testing team is not involved in the system testing. Achieving accord between the internal and external testing teams at the beginning and the end of the development stages is always difficult and seldom lasting.

According to another arrangement the 3rd party fully assumes testing responsibilities for the system. In this case a formal testing stage is often substituted for an informal procedure with developers executing ad-hoc tests on each other's code. More advanced development agencies hire internal test specialists or outsource testing to one or more testing companies.

2.1.2 Outsourcing of testing work

With complexity and diversity of technologies increasing, e-testing becomes more specialised. For instance, functional testers and security testers are likely to need

different set of skills. The same goes for load -, log - and usability testers. Along with the deeper specialisation of services we also notice appearance of commoditised testing functions. On-line companies offer usability or load assessments. Specialisation and commoditisation increase the role of outsourcing in testing. Often, several companies specialising in different areas test the same product. Assuming there are enough service providers, the outsourcing solution is appealing. It decreases costs to a development company and eliminates the risk of failure [4], [5].

2.2 Externalisation of internal systems and the consequent changes in the operating requirements

The scope and reach of web-based applications have greatly increased, comparing to those of traditional systems. Customers, from in-house representatives changed to external users. Thus, the externalisation resulted in much higher operating requirements than requirements applied to traditional information systems (IS). Under operating requirements I designate guaranteed up-time, access requirements, performance and volume conditions, supported languages and channels. Among the examples are 7 by 24 support, access from different time zones, system capacity for peak hours/seasons, different client hardware/software configurations, and multi-lingual support.

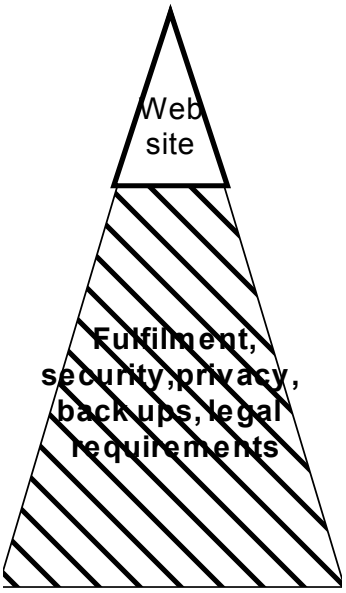
Changing operating requirements mostly affected load testing, performance testing and system comparability testing. Other examples are testing in 'odd' hours, testing in different languages which often done on specific keyboards (Japanese, for example).

2.3 Changing role of web systems - the iceberg effect

A web site, which appears to a user on Internet, in fact, is only a tip of an iceberg. There are various fulfilment, procurement, legal, etc. services enabling a web site to function. Internet-based products handle interactions with the customers (customer support role), represent a company to the public (public relationships role), and offer certain service-related guarantees to the customers (compliance role). Testing, as checking the quality of a web site assumes new flavours. While traditional testing checks the pure software development categories (functionality of a product, load and performance characteristics and security), the e-testing often borders with marketing, internal compliance and auditing procedures. Some examples include verification of privacy procedures (auditing, compliance); answering customers' complains within the guaranteed time frame (customer support), usability assessments and post-installation log analysis (marketing).

In traditional systems testing had its stable place among marketing, compliance, and development departments. Now many roles shifted and testing departments have taken to carry more on their shoulders. Involvement of the testing departments in these new roles is quite understandable because, e-systems have originally developed within development departments. Testing departments became involved on early stages as well. Unlike compliance, marketing and other business departments which did not interact with new technology until recently. Gradually, the progress of establishing and institutionalising the internet channel will position marketing and compliance functions where they belong, once internal marketing and compliance departments will get more familiar with the technology and assume their share of responsibilities in e-systems.

Figure 1: The iceberg effect

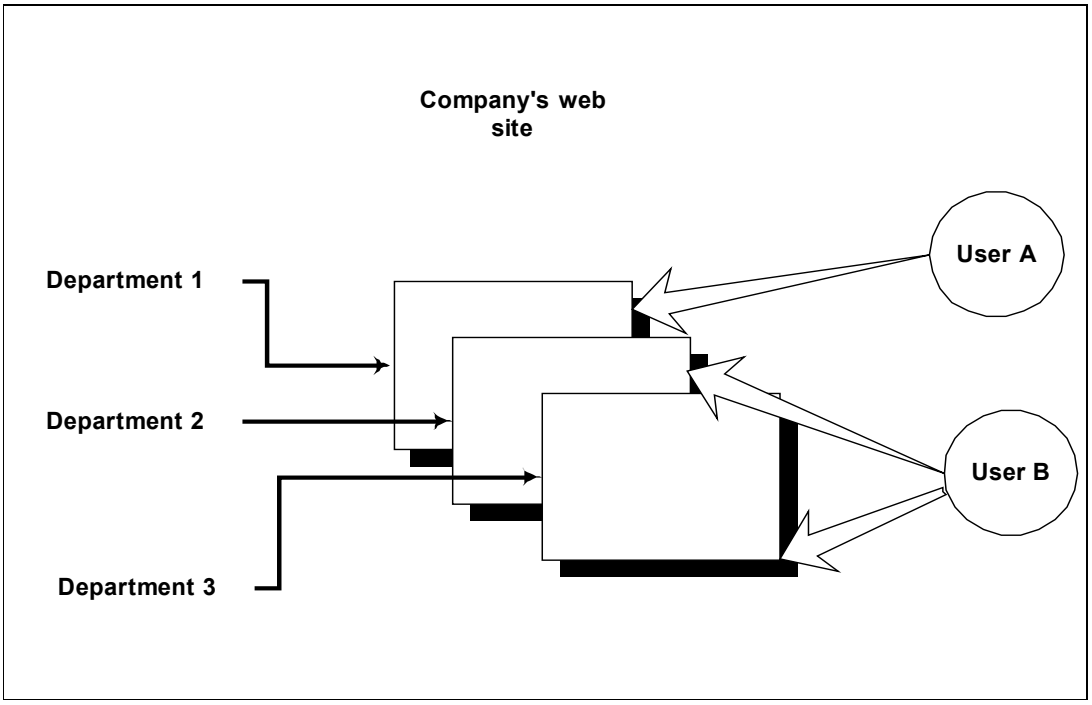


2.4 System and media complexity

Papers [2] and [3] argue that the development of web information systems requires a rather different approach from that of traditional IS development methodologies. They attribute this to the tight integration needed with other information systems, such as databases and transaction processing systems, as well as the much wider audience addressed by these systems. This technical and business integration is required both inside the host organisation and with the organisation's trading partners; in other words, it needs to fit the inter-organisational situation.

Often, a single web information system represents functionality provided by multiple internal departments. It may serve as a single platform for these departments to interact with external customers. In this case the system has to be thoroughly tested for integration among all internal players. Similarly, the system has to deal with different segments of customers and so must allow for a group - or individual customisation. This multi-layer customisation has to be tested as well. To complicate issues mentioned above, companies have to enable web sites with multi-media (sounds/voice, animation, graphics) and now SMS and WAP functionality. Integration among the delivery channels also has to be tested.

Figure 2: An internet-based system often serves as a single platform for interactions between multiple categories of users and multiple departments within a company.



3 Consequent changes in the testing process

All the described changes reshape methods and techniques used in e- testing. E-testing assumes new roles and adopts new approaches. Below, I catalogue the high-level changes required to adapt traditional testing business to an internet environment.

Tables 1: Changes in e-testing

Category of a change	Changes in e-testing process
Software life cycle changes	1. Increased attention to user acceptance testing. Users involvement in requirements generation. 2. Increased attention to integration testing and integration co-ordination challenges.
Externalisation of the systems and consequent changes in operating requirements	1. Increased attention to performance and load testing. Usage of load stimulators. 2. 24 x 7 testing, testing during 'odd' hours. 3. Increased attention to security testing. 4. Comparability testing (cross-platform, cross-browser, different screen sizes and resolutions; text-only mode). 5. Test access for disabled people. 6. Test in multiple languages, currencies and measurement systems. 7. 'Fool-prove' testing (a product has to work for anyone, not for experts - like a TV-set). 8. Increased importance of testing of on-line Help. User manuals change to shorter on-line Help descriptions that need to be tested for thoroughness and completeness. 9. 'Exploratory testing'. Users do not have manuals explaining how to use a product, at the best, they have on-line help; users explore a site. Therefore testing should mimic their behaviour and include exploratory testing in the portfolio of services. 10. Usability testing. 11. Personalisation testing, testing for multiple categories of users.
System and media complexity	1. Growing importance of regression testing and change management control. 2. Extensive page layouts, video and audio tests. 3. Test multiple/alternative delivery methods/technologies (video, sound, WAP, SMS).
Changing role of web systems - the iceberg effect	1. Post- installation testing (log analysis to identify user's patterns and system glitches preventing them). Testing borders with marketing. 2. Quality and security audits. 3. Test to ensure that the virtual world meets the material one: external customers have to be sure that the system is sufficient to achieve what it is claiming to achieve (handle orders, delivery time - warehouses/3rd parties involved). 4. Ensure that external customers are protected from the system (no fraud, privacy and security issues, back up procedures). 5. Test in comparison to competitors: to work - is not enough, need to work at a certain 'standard' level. Obtain seals of quality approval.

4 Future trends

I outlined changes in the testing methods and techniques caused by the differences between on-line and regular systems. I see the growing importance of e-testing being supported by:

- The maturity of the technology and users community -
  With the Internet community and relevant technology maturing, quality of web systems will become more important. Quality of web sites will become a differentiating factor between the competitors.

- Growing role of on-line applications in revenues -
  Growing role of on-line applications in revenues increased the risk of company failure by not paying attention to a quality of web applications. Thoroughness and amount of testing acceptable for a small channel is not acceptable for a major revenue producer.

- The maturity of the industry standards and of the governing law -
  The maturity of the industry standards and of the governing law will force companies to pay more attention - and therefore apply more testing - to such areas of web development as privacy of data, provision of the related information and access for disabled users.

I believe we will soon notice the emergence and institutionalisation of uniform testing standards in the e-testing area. There are companies currently working on standards in security, content rating, usability, and web audit areas.

The more work is probably required to test validity of dimensions I have offered to describe changes in testing business. So far my practical experience proved that operating in these 4 dimensions enabled us to build a successful e-testing business

References

[1.] R. Clarke, The Conventional System Life-Cycle Xamax Consultancy Pty Ltd. *http://www.ana.edu.au/people/Roger.Clarke/SOS/SLC.html*, 1997

[2.] Isakowitz T., Bieber M. and Vitali F. (1998) Web Information Systems, *Communications Vol. 41, No. 7.*

[3.] D. Klyachko and S. Smithson, A Strategy for Web Development in Electronic Trading: A case study. Twelfth International Bled Commerce Conference, Bled, Slovenia 1999.

[4.] M. Lacity and L. Willcocks, "An empirical investigation of information technology sourcing practices: lessons from experience" in *MIS Quarterly* vol. 22 Number 3, September 1998.

[5.] M. Lacity and R. Hirschheim, Information systems outsourcing: myth, metaphors and realities. Wiley, 1993.

[6.] E. Yourdon, Modern structured analysis, Prentice-Hall, 1989

Specifics of e-testing: difference between traditional and on-line software development and its effect on testing

Dasha Klyachko
Allied Testing, LLC
PhD researchers at the London School of Economics
dasha@alliedtesting.com

Categories of changes between traditional testing and testing of on-line applications

1. Software life cycle and project structure changes
2. Externalisation of internal systems and sequential changes in operating requirements
3. Changing role of web systems - the iceberg effect
4. Media and system complexity

Category of a change

Software life cycle (SLC) and project structure changes

Details

- SLC changes
- Outsourcing of development
- Outsourcing of testing work

Changes in testing

- Increased attention to user acceptance testing. Users involvement in requirements generation.
- Increased attention to integration testing and integration co-ordination challenges.

Category of a change

Externalisation of internal systems and sequential changes in operating requirements

Details

- Guaranteed up-time
- Access requirements
- Performance and volume conditions
- Supported languages/channels

Changes in testing (page 1)

- Increased attention to performance and load testing.
- 24 x 7 testing, testing during 'odd' hours.
- Increased attention to security testing.
- Comparability testing (OS, browser, screen size and resolution)

Changes in testing (page 2)

- Test access for disabled people.
- Test in multiple languages, currencies and measurement systems.
- 'Fool-prove' testing.
- Increased importance of testing of on-line Help.

Changes in testing (page 3)

- Exploratory testing.
- Usability testing.
- Personalisation testing, testing for multiple categories of users.

Category of a change

Details

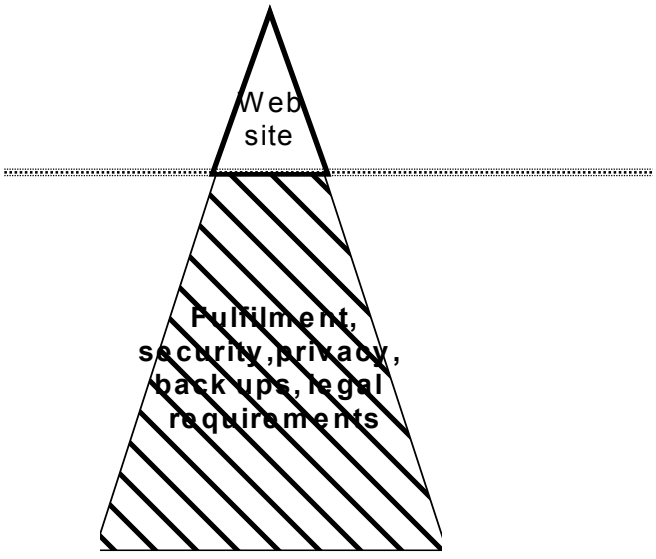
Changing role of web systems - the iceberg effect

•Fulfilment, procurement, and legal requirements.

• Interactions with the customers.

• Represent a company to the public

• Offer certain service-related guarantees to the customers



Changes in testing (page 1)

- Post- installation testing (log analysis).
- Quality and security audits.
- Test to ensure that the virtual world meets the material one (order processing, delivery time, 3rd parties, fulfilment).

Changes in testing (page 2)

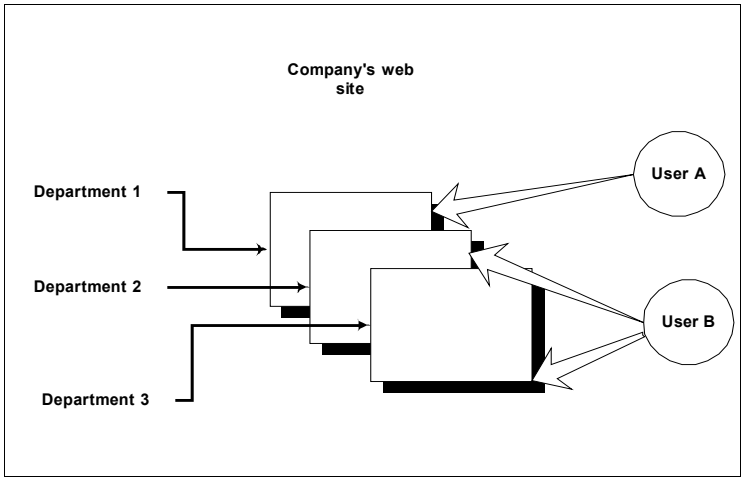
- Ensure that external customers are protected from the system (no fraud, privacy and security issues, back up procedures).
- Test in comparison to competitors: to work - is not enough, need to work at a certain 'standard' level.
- Obtain seals of quality approval.

Category of a change

Media and system complexity

Details

- Single platform for multiple
- Interactions with multiple customers' segments
- Multi-media (text, sounds, animation, graphics)
- Multi-channels (web, SMS, WAP)



Changes in testing

- Growing importance of regression testing and change management control.
- Extensive page layouts, video and audio tests.
- Test multiple/alternative delivery methods/technologies (video, sound, WAP, SMS).

Future trends

- Growing importance of e-testing
  - The maturity of the technology and users community
  - Growing role of on-line applications in revenues
  - The maturity of the industry standards and of the governing law
- Appearance of the uniform e-testing standards.

QWE2000 Session 9I

Mr. Ardian Cowderoy [UK]
(ProfessionalSpirit)

"Technical Quality Is Just The Start -- The Real Battle Is Commercial Quality"

Key Points

- System Quality
- Quality in Use
- System Benefits

Presentation Abstract

In 1997 and 1998 a consortium of European companies and universities, supported by the European Commission, undertook a major exercise to explore the meaning of quality in the context of multimedia and website development, with attention to both the content and functionality. The author was the project manager and technical director of this research project, "MultiSpace".

Technical quality. Each of the different digital assets used in websites and multimedia has distinct characteristics that asset developers believe to represent "quality", such as listed in Table 3 below. While the importance of some technical quality features have loose relationship to system quality, others can be critical. For example, colour accuracy is safety-critical for some medical and defence systems.

The MultiSpace project reviewed system quality features, and especially those of the draft ISO/IEC 9126.1 and .2 ("Software product evaluation - Quality characteristics and guidelines for their use"). The project found that the ISO/IEC 9126 characteristics and sub-characteristics can be interpreted for multimedia and website content in a similar way as multimedia and website functionality. Indeed, the two are sometime inseparable (such as in usability).

However there were some significant technical problems. The first was that the usability sub-characteristic of "attractiveness" was inadequate at describing the richness of the experience of using multimedia and the Internet. The second was that the entire topic of usability concerns only reducing pain of using the human-system interface, and there is minimal attention to the joys. An effective description of website/multimedia quality would also be effective at describing the quality of the component media: books, films, training, games, advertising, etc.

Quality in use. Software systems are developed for a use (or set of uses) resulting from clearly defined needs. Although some multimedia and websites are utilitarian,

many are also designed to please personal needs. Especially in the context of websites, these personal needs comes from a highly diverse audience and are difficult to model.

We have found it useful to make a distinction between the system constraints needed for achieving the primary purpose, and the system benefits that create the experience.

System benefits. System benefits involve giving people more than they expected, in a satisfying way. Some vendors of commercial software packages achieve this by adding a wide range of new functions, some of which may be useful. Apart from resulting in bloatware and increased maintenance costs, this focuses only on utilitarian purpose. In contrast, traditional media combine quality with a heavy emphasis on rewards that create increase the quality of the experience. Specifically, they offer a variety of novelty features, supplementary learning, inter-personal participation and emotional satisfaction (or stimulation). Table 4 below lists sub-characteristics that describe these effects.

About the Speaker

Adrian Cowderoy is Managing Director of the Multimedia House of Quality Limited, a company which he established to promote quality-improvement methods for the production of websites and multimedia.

Mr Cowderoy was the General chair of ESCOM-SCOPE-99 and ESCOM-ENCRESS-98 conferences, and was Program chair for ESCOM 96 and 97 (The European Software Control and Metrics conference promotes leading-edge developments in industry and research, worldwide û see www.escom.co.uk). He is the METRICS-ESCOM Coordinator for IEEE METRICS 2001 and was on the Program committee of Metrics 98 and 99, European Quality Week 99 and COCOMO/SCM 96-99. In 1998 he was acting Conference Chair of the Electronics and Visual Arts conference in Gifu, Japan. He is a registered expert to the European Commission DGXIII.

He has provided consultancy and industrial training courses on quality management, risk management, and cost estimation to the aerospace and medical industries in the UK, Germany and Italy since 1995. He also lectures at Middlesex University (www.mdx.ac.uk) on e-commerce project management and managing Internet start-up's, and at City University, London (www.city.ac.uk), on project management for systems development.

Mr Cowderoy was project manager and technical director of MultiSpace, a 14-month million-dollar initiative sponsored by the European Commission in which 12 European organizations explored the potential to apply quality-improvement methods to multimedia and website development projects. (See www.mmhq.co.uk/multispace and www.cordis.lu/esprit.)

He was a Research fellow at City University from 1990-1998, and a Research Associate at Imperial College from 1986-1989. He was also a quality consultant and

software developer at International Computers Limited, UK, from 1980-1985, where he worked on operating and networking systems for mainframes and distributed systems.

His academic qualifications include an MSc in Management Science from Imperial College, University of London in 1986, and is a member of the Association of MBA's. He received a BSc in Physics with Engineering from Queen Mary College, University of London, in 1979.

Mr. Cowderoy has published and presented extensively on multimedia quality and software cost estimation. He was joint editor of Project Control for 2000 and Beyond (Elsevier, 1998), Project Control for Software Quality (Elsevier, 1999), and Project Control: The Human Factor (Elsevier, 2000).

PROFESSIONAL *Spirit*

Technical Quality is just the start – the real battle is *Commercial Quality*

Adrian Cowderoy

PROFESSIONAL *Spirit*

What is meant by Quality?

- The MultiSpace Project –
  - Major initiative supported by European Commission to explore the application of quality engineering techniques to multimedia and the web
  - 12 companies and universities
- Questions:
  - Is content different to functionality?
  - Do software quality characteristics describe books, teaching, music and entertainment?
  - How can quality be specified, tested and managed?

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 2

PROFESSIONAL *Spirit*

The Contract View: System Quality

- Externally visible view of quality
  - allows effective management of contracts
  - is adaptable to "content"
    - Ch.3 of http://www.mmhq.co.uk/multispace/d2-2p.pdf
- Example, ISO/IEC 9126.2
  - Functionality (suitability, accuracy, security, etc)
  - Reliability ()
  - Usability (learnability, understandability, operability, attractiveness)
  - Maintainability ()
  - Efficiency ()
  - Portability ()

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 3

PROFESSIONAL *Spirit*

Example measure of system quality

"Using text input"

X = A / B

A = number of places where text input can be used
B = total number of places where input can be given

Interpretation:

$0 \leq X \leq 1$

"A higher value means that more text input can be used."

Measures

X is ratio-scale metric, A & B are counts

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 4

PROFESSIONAL *Spirit*

The Developer's View: Technical Quality

- Asset quality (visible to users)
  www.mmhq.co.uk/my-quality/measure-content-quality.shtml
  - Narrative
    - Spelling, grammar, linguistic simplicity, tone of voice, compactness, words per section, logical structure, text layout
  - Images 2D
    - Image quantity, image compression, contrast control, gradation, edge sharpness, use of color, color mode, color fidelity, printing accuracy, color vibrancy, harmony with scheme, composition.
  - Navigational aids, Movies, 3V & VR, etc
- Use for specifying testing critical features
  - e.g. medical and cultural arts need quality accuracy
- And use for improving task performance

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 5

PROFESSIONAL *Spirit*

Other types of Technical Quality

- Asset complexity (internal characteristics)
  http://www.mmhq.co.uk/my-complexity/
  - object complexity
    - e.g. image layers, size, colors, etc
  - structural complexity between objects

  → indicates possible problems:
    - identify high-risk components
    - charge-rate for changes, subject to constraints
    - indicators of poor operability
- Process efficiency
  - asset management system
  - documentation

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 6

PROFESSIONAL *Spirit*

The Commercial View: System Rewards

- Usability: attractiveness
  - usability is the efficiency of human-machine-interface
  - only "sufficient" usability is needed
- The web involves multiple media
  - what makes a good book, film or music track?
  - what makes effective training?
  - what increases sales?

→To compete with the top web-developers
ALL quality perspectives must be considered

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 7

PROFESSIONAL *Spirit*

The Commercial View: System Rewards

- System constraints
  - need just enough for users, immediate business purpose, and long-term maintainability
- Technical quality of assets
  - need just enough for critical assets
- System rewards
  - effects that enhance business purpose
  - need as much as possible within project constraints

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 8

PROFESSIONAL *Spirit*

System Rewards

- 5 characteristics (with 29 sub-characteristics):
  - Participation (…, ...)
  - Discovery (…, ...)
  - Novelty (experience, paradox, dramatic, technical)
  - Involvement (…, ...)
  - Comfort (…, ...)
- Balance
  - different people like different features
  - some have negative effects
  - some features work best together
  - some features are contradictory

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 9

PROFESSIONAL *Spirit*

Sub-characteristics of Novelty

- Personal experience
- Paradox/riddle
- Dramatic experience
- Technical gimmicks

BEWARE:
Balance novelty with Usability and Comfort

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 10

PROFESSIONAL *Spirit*

Sub-characteristics of Discovery

Learning about …

…people

…events

…organizations

…tools and processes

…science and technology

…the world

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 11

PROFESSIONAL *Spirit*

Sub-characteristics of Participation

- Bulletin boards
- Email circulars
- Real-time interaction
- Physical meeting support
- Pushed content
- Personalization of interaction

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 12

PROFESSIONAL *Spirit*

Sub-characteristics of Involvement

Encourage selected emotions
and discourage others

- News media often rely on anger, fear, pity
- Advertising relies on confidence, greed/avarice
- Entertainment exploits opposites
  (e.g. love and hate)
  and excites physiological effects (such as lust)

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 13

PROFESSIONAL *Spirit*

Sub-characteristics of Comfort

- Visual beauty
- Poetic act
- Engineering elegance
- Expectation
- Conformance to social norms

N.B. Comfort is a reward,
Usability is a requirement

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 14

PROFESSIONAL *Spirit*

Using benefits: Resolve conflicts

- Resolve conflicts
  - business and marketing people need cost control
  - users want as many benefits as possible, and sufficient usability, etc.
  - technical staff have technical constraints from the tools and legacy content
  - domain specialists message constraints (for training, advertising, etc)
- Proposed mechanism
  - rank each (relevant) feature on scale 1(irrelevant) to 3 (critical)
  - use spreadsheet to identify differences
  - consider costs, risks and benefits of each feature

http://www.mmhq.co.uk/quality-planning/

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 15

PROFESSIONAL *Spirit*

Using benefits: User diversity

- Linguistic ability
- Familiarity with the domain
- Computer literacy
- Web-behavior
  - seeker, resident, tutor, etc …
- Cultural
  - different learning methods
  - different problem-solving methods
  - different interpersonal behavior
  - different value systems
  - different history (and symbols)

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 16

PROFESSIONAL *Spirit*

Using benefits: Plan Quality

1) Is it engaging?
   – it takes one click to leave a web-site !
2) Is it inspiring?
   – why return to the website?
3) Is it cool? (Is the style right?)
   – being cool is hard work.
4) Is it effective?
   – are the promises fulfilled?
5) Is it maintainable?
   – adaptations and extension often cost more than the original website creation.

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 17

PROFESSIONAL *Spirit*

Is it cool?

- Cool is being laid back
  – cool websites do not require massive content or functionality (although they may have them)
- Cool is being self-assured
  – your website mustn't pretend to be something else
- Cool is being consistent
  … until you change your mind
- Cool is about style, not functional or utility
  – some cool sites are difficult to use
- Coolness can not be photocopied !
  – as soon as it is defined, it becomes functional

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 18

PROFESSIONAL *Spirit*

Examples of building coolness

- Novelty
  - personal experience - create experiences for visitors or describe cool experiences, but stay laid-back
  - paradox/riddle can be used to create coolness - take care to maintain consistency
  - dramatic experience requires a great skill to do in a cool way - don't photocopying other people
  - technical gimmicks - avoid gimmicks that are merely functional or are widely used
- Likewise for discovery, participation, involvement, comfort
- Usability, reliability & performance must be OK

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 19

PROFESSIONAL *Spirit*

Using benefits: Control Asset wastage

- Content from libraries and subcontractors can dominate project costs
- Cost of errors
  - unsuitable content has to be changed or replaced
  - indirect costs of replacing material in the field
- Impact of changed priorities
  - cost of new content, and waste when some old content is abandoned
  - increased opportunity and revenue-earning potential

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 20

PROFESSIONAL *Spirit*

Process improvement - how?

- Manufacturing processes: reduction of waste
  - equates waste with inefficiency
  - benefits from statistical process control
  - needs process standardization
  - relies on steady changes in technology and market
- Alternative improvement mechanisms
  - risk reduction
  - quality checking at milestones
  - performance review based on 2-4 measures
  - improved selection of producer/director
  - improved stimuli and resources (for creativity)

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 21

PROFESSIONAL *Spirit*

Web sites

http://www.mmhq.co.uk/

/my-quality/measure-it.shtml … quality measures
/my-quality/is-it-ok.shtml … applying measures

/multispace/framework.shtml … quality processes from the EU project

/my-complexity/ … complexity measures
/quality-planning/ … mini-tutorial

Related topics

http://www.emmus.org/ … web usability
http:/www.cordis.lu/ … EU research projects

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 22

PROFESSIONAL *Spirit*

Contact details

Adrian Cowderoy

email adrian.cowderoy@professionalspirit.com

phone +44(UK) 118 9 427 970

snailmail ProfessionalSpirit
46 Western Alms Avenue
Reading RG30 2AN
United Kingdom

"Technical Quality is just the start – the real battle is Commercial Quality" QWE2000, page 23

More on web quality
www.mmhq.co.uk

Technical Quality is just the start.
The real battle is *Commercial Quality.*

Adrian Cowderoy

PROFESSIONAL *Spirit*

In 1997 the European Commission formed a project to explore the *meaning of quality* in the context of multimedia and Internet system development, and to establish whether engineering approaches to quality-improvement could be applied to this emerging domain.

The starting point were the various software frameworks used for quality specification, including the perspectives of system quality, quality in use and Technical Quality ("internal product characteristics"). We anticipated that the perspective of Technical Quality would need massive extension to support individual digital assets (pictures, text, movies, etc). What we did not anticipate, was that that there is a fourth perspective on quality that has been almost entirely ignored in the software industry.

This paper is about the fourth perspective, and how it was found.

1) Common perspectives on quality

The MultiSpace project combined experts from twelve organizations in the software quality industry with website developers and experts. (The term "website" is used in this paper, but most of the material also applies to multimedia, Internet, intranet and extranet systems.) The first meeting was traumatic – the software people had tightly refined ideas about quality which the web-development people saw as seemingly irrelevant to most of the real issues they regularly faced in making effective systems.

The first exercise was to examine the existing materials and listen to the needs of a wide variety of practitioners. With this the team developed its many initial views into a single synthesis that considered what quality meant, how it could be managed and how it fitted within the website development cycle. (There would be little point in defining quality, if it could not be managed.) It also considered how to profile the very diverse range of users. Collectively, this represents the MultiSpace Framework [Cowderoy & Daily, 1997].

Amongst the observations in the MultiSpace Framework are that quality is important for managing contracts, for planning the quality-improvement and testing, and for evaluating the different responses that users have to a single system. For this, descriptions are needed of system quality, Technical Quality, and the users. This section presents the first two of these – the third is presented in section 2.5.

1.1) System Constraints

Effective management of contracts is essential to client-contractor relationships when the financial amount is to great for relationships of trust, or when the technical complexity is too great for an individual architect, or when a market-driven approach is used for production of individual components. (Market-driven production involves outsourcing each step of the production chain to whichever contractor is cheapest at the required quality and delivery time [Williamson 1983].)

Efficient contracts for website development need to specify what needs to be delivered in terms of content, functionality and quality. The quality specification needs to address all quality characteristics, not just those that are easy to define.

In the software industry, various schemes have been produced for measuring the quality of delivered functionality. Most of these focus on measuring a specific characteristic, such as reliability or usability, and ignore other issues. Unfortunately schemes often contradict each other. For some years, the ISO/IEC 9126 committee has been working on a quality characterization scheme that describes all quality characteristics and sub-characteristics of software. This also has contradictions with other schemes, and weaknesses (such as with safety issues), but it forms a useful starting point for describing quality. (The 10 Design Quality features proposed by Fellenstein and Wood of IBM [2000], also correspond to a subset of 9126 quality sub-characteristics.)

Websites include deliver both content and functionality. The functionality is described by the existing 9126 definitions. In the MultiSpace project, we experimented with adapting those definitions to describe content. This included text, 2D images, 3D images, video, audio and navigation aids. The subcharacteristics descriptions can be found in chapter 3 of the MultiSpace Priorities [van Veenendaal]. These are supplemented by over 100 measures, with support for all the subcharacteristics [Boonstra, 1998]. When MultiSpace started, the accusation had been labeled at the project that quality was far too illusive to be measured. The work of the Philips and KEMA teams in MultiSpace proved that content quality *can* be measured for websites, in the context of fitness-for-purpose.

Table 1. Examples of Operability metrics for Internet and multimedia systems [Boonstra, 1998].

External metrics	**Measurement**	**Interpretation**	**Scale type**	**Measure type**
Using text input	X = (A / B) A = number of places where text input can be given B = total number of places where input can be given	0<=X<=1 A higher value means that more text input can be used.	Ratio	X = Ratio A = Count B = Count

Using voice input	X = (A / B) A = number of places where voice input can be given B = total number of places where input can be given	0<=X<=1 A higher value means that more voice input can be used.	Ratio	X = Ratio A = Count B = Count
Using displacement input	X = (A / B) A = number of places where displacement input can be given B = total number of places where input can be given	0<=X<=1 A higher value means that more displacement input can be used.	Ratio	X = Ratio A = Count B = Count

1.2) Technical Quality

Each of the different digital assets used in websites has distinct characteristics that "quality", such as listed in Table 2 below. The extent of the contribution to quality varies between applications, from extremes of zero relevance to extreme high. For example, color accuracy is safety-critical for some medical and defense systems, textual accuracy is system critical for web-retail sites and teaching materials.

Table 2. Summary of characteristics for different types of digital asset.

Asset type	**Characteristics**
Narrative	Spelling, grammar, linguistic simplicity, tone of voice, compactness, quantity per section, logical structure, text layout.
Navigational aids	Dynamic effects, navigation harmony with main scheme, hyperlink quantity, URL addresses shown, exclusion of advertising banners.
Images 2D	Image quantity, image compression, contrast control, smooth tonal transition ("gradation"), edge sharpness, use of color, color mode, color fidelity, printing accuracy, color vibrancy, image harmony with main scheme, image composition.
Movies	Frame size, movie compression, frame rate, directing, movie composition, editing, after effects.

For detailed descriptions, see http://www.mmhq.co.uk/my-quality/

There are also suggestions that the adaptability and extendibility of a website may be linked to complexity and size of individual assets, and the compound structures that combine assets [Cowderoy, 2000 and http://www.mmhq.co.uk/my-complexity/benefits.shtml].

2) System Rewards

2.1) The argument for System Rewards

For centuries, engineers have worked to satisfy the needs of their clients and the public. In past centuries, this was sometimes supplemented by flights of creativity that added features that were aesthetically pleasing to some people, but served no clear functional purpose.

During the twentieth century, the influence of factory processes was extended to other industries. This created a massive improvement in the standard of living, especially as a result of the widespread use of statistical product and quality techniques that support ever-closer satisfaction of functional purposes. With such techniques, we could now construct the architecture of St Peter' s dome, the works of Shakespeare and the music of Beethoven with much greater efficiency – surplus ornamentation, words and notes would be removed, achieving improved value for money and higher system efficiency.

The importance of beauty was recognized by the ISO/IEC 9126.2 committee when they defined the usability sub-characteristic of Attractiveness. However in its current form it is inadequate at describing the richness of the *experience* of using websites.

The ISO/IECE 9126.2 use of Attractiveness is also in contradiction to the other usability sub-characteristics, which describe how the *pain* of using the human-system interface can be reduced.

An effective description of website quality would also be effective at describing the quality of the component media: books, films, training, games, advertising, etc. if Michelangelo, Shakespeare and Beethoven were correct in their works, then something is missing from the engineer' s definition of quality. Morespecifically :-

> *The commercial failure of some websites can be linked to bad system quality, but once adequate system quality has been achieved, there is no structure for defining what makes some sites a commercial success, and others a disaster.*

Following some earlier work in MultiSpace, we proposed at the IEEE Symposium on Software Metrics that, with some exceptions, websites have to provide rewards to their users as well as satisfying needs [Cowderoy *et al*, 1998]. In effect, quality consists of two sets of features:

- *System Constraints* involving the satisfaction of needs to perform the functional purpose of the website. A well-designed system provides "just enough" quality for the purpose – anything more would reduce productivity and delivery time.
- *System Rewards*, involving the provision of features in addition to the primary purpose. Such features are individually dependent – each person likes some features, and ignores (or dislikes) others. The objective of the developers is to provide as much reward as possible within the budget, and timescale. Rewards increase attention, prolong visits to websites, and encourage return visits. The financial benefits of these can be so significant that it can justify increased budget and delayed delivery.

The system benefits are listed in detail in section 3 and section 4.

2.2) Asset wastage

Ideally, System Rewards should be defined at the start of the project, and either included in the budget or prioritized so that the most important ones are included. However typically System Rewards can only be outlined at the start of the project in a very general fashion. The specific ideas only come part way through, and may come at the first customer demonstration. These late arrivals may require a change of priorities partway through the project. An effect of changing priorities is a level of waste – i.e. assets that are partly developed, then discarded.

There is a misleading similarity between the wasted resources from discarded assets in content development, and the wasted effort resulting from late detection of faults in software development. The main differences are :-

- Wasted assets can occur from *changed priorities*, resulting in increased financial advantage.
- Wasted assets can also occur because the design includes some design features that are certainties, and others that are *opportunities* – i.e. there is a possibility that their impact will be different to original expectations (greater or less), and they may create secondary risks (such as reduced usability and reliability and new opportunities.)

2.3) Quality improvement strategies

In software engineering, it is often said to be "good practice" to use processes that reduce the possibility of faults. However in content development, the major challenge is to manage the risks and opportunities efficiently. The most efficient way of handling this is with risk management techniques, but to list both good and bad events, and control actions that create leverage to improve each opportunity or risk. Often these control actions correspond to conventional quality assurance and quality control actions, however compared to standardized software engineering practices, these extra quality methods are used only when the benefits clearly outweigh the costs for the current project.

Another promising methods for web development is concurrent engineering [Carter & Baker]. The concurrency helps with Rapid Application Development, the massive broadband communications helps improve creativity and reduces many of the quality risks within the project. It also supports risk management, quality profiling and people management,

2.4) Introducing quality-improvement techniques

We have encountered some hostility to the introduction of engineering methods into environments that depend on creativity and craft skills. This has led to various conclusions:

- Process definition, standardization and inspection techniques currently only apply to certain activities.
- Quality engineer' s belief in the importance of statistical methods is seen as a religion (sometimes fanatic) and not a proposition that is feasible in an industry that can see major changes within only 3 months.

- Quality-training needs to focus on a few key issues to which web development staff can directly relate.

Table 3. Example of a teaching aid in this area
from http://www.mmhq.co.uk/cool-way/

To teach newcomers to project management, the MMHQ website encourages people to focus on four areas: the project participants, the product quality, the project risks, and control mechanisms. (In this context, control is used to improve communications, quality and risks. The strong emphasis on project participants is important for a craft-based industry.) The emphasis is on increasing awareness of how each of the four areas effects the others, and can be used to improve the others. Within each of the four areas, people are encouraged to appreciate the breadth of topics that should be considered. To highlight interest, the exercise is presented as a set of card games ("the Cool Way"). It encourages use of (conventional) methods such as risk management and quality profiling, and it provides a framework on which more detailed and advanced methods can be built.

2.5) User perspectives

Content-rich web-sites may receive a much more diverse range of users than conventional software systems.

- For software, the designers and usability evaluators have to address different levels of computer literacy, physical/visual dexterity and (perhaps) familiarity with the application area.
- For websites, the "familiarity with the application domain" can become much more complex, especially for the System Benefits. There are also important differences from web-behavior (seeker, resident, tutor, etc), linguistic ability, national variations in group behavior and learning strategies, and value sets (such as from ethnic traditions). Failure to adequately address these differences can result in customers abandoning the web-site, and can even result in hostile behavior.

To perform statistical evaluation of the system rewards websites would require a large population, sampled to include all these different categories. This would be expensive, time-consuming and it can be difficult to maintain confidentiality an alternative is to use a small sample, that are carefully chosen so that each category is represented by one (or more) people in the sample. The people in the sample tend to fit into three groups :-

- *Novices* are new to evaluation, and give unbiased opinions, which need to be solicited using a combination of questionnaire, interview and observation – i.e. the evaluation cost is relatively high.
- *Actors* are experienced evaluators who play a specified role – they may give very detailed responses, responding using questionnaires and freestyle, but there is risk that their responses are stereotyped.
- *Focus-group members* interact with each other (as well as the moderator) may tend to double-guess business strategies and other information that has been withheld from them.

3) Planning the System Rewards

There are five general categories of reward. Existing websites often totally ignore some of these categories, but in commercially-oriented websites, most or all of these need to be considered.

- *Participation* is the core of the one-to-one marketing and community-based e-business which is one of the characteristics of the New Economy [Siegel, 1999].
- *Discovery* increases retention and, if there is sufficient new material, it encourages return visits.
- *Novelty* attracts attention, which increases retention at the site and may improve concentration.
- *Involvement* increases concentration and retention. When there is a strong emotional element it can encourage return visits.
- *Comfort* is essential for people to cope with the stress from using the system and its rewards.

The 5 reward categories subdivide into 29 specific features. Effective quality planning needs all 29 specific features. In assigning importance, beware of the special cases:

- Some features may have negative effects for a particular business or audience. (For example, most sites would be negatively influenced by the inclusion of any feature than incited lust.)
- Some features are complimentary to other features within the same category. These work better when used together. (For example, "poetic act" benefits from co-existence with "visual beauty".)
- Features in one category may compliment or clash with features in another category, or with System Constraints. (For example, novelty requires the presence of comfort and the system constraint of understandability.)

As well as considering special cases, the importance of each feature needs to resolve conflicting needs from the four types of player in the project:

- *Business and marketing people,* who require the website to justify its costs. To this purpose, they need the inclusion of some features but see no direct financial benefit from many others, unless these other features change the behavior of users.
- *Users,* who may include different and conflicting audiences and may include people who are hostile to the website owners. As explained in section 2.5, there is usually a diverse range of users to interview.
- *Domain specialists,* who protect the site from misrepresentation of messages. (For example, marketing people protect brand images, and teachers demand strict use of terminology.)
- *Technical staff,* who carry the legacy of the existing website, the constraints from tools, and the constraints of existing software solutions to which the web-site provides a front-end (which is common for advanced e-commerce systems).

It takes a modest effort to identify the priorities for so many players, and then resolve them, (A set of linked spreadsheets helps.) However without that effort there is a risk that key features have been forgotten, or under-emphasized, or exaggerated. The cost of such mistakes can be much greater than the effort of performing quality profiling.

4) Characterization of System Rewards

This section contains lists of the various characteristics and sub-characteristics which collectively describe the System Rewards. Examples can be found at http://www.mmhq.co.uk/my-quality/examples.shtml

4.1) Participation

Participation includes involvement with other people in both real and virtual contexts. On the web, participation tends to be aimed at very specific user communities (i.e. people with some similar objectives). To make participation effective other reward features may be need to emphasized to suit the local needs of the target users.

N.B. Participation often requires major technical challenges, which introduce other problems (such as with reliability, usability and performance).

Feature	Summary	Simple scale
Bulletin boards	Electronic notice-boards to which user's can append their own messages (usually without censorship).	Irrelevant, Minimal, Good practice, or State-of-art.
Email circulars	Letters sent electronically to an distribution list.	Irrelevant, Minimal, Good practice, or State-of-art.
Real-time interaction	In which user's engage in real-time dialogue in an on-line environment. (Includes use of avatars and video conferencing.)	Irrelevant, Minimal, Good practice, or State-of-art.
Physical meeting support	Support of meetings with preparation, broadcast, and dissemination.	Irrelevant, Minimal, Good practice, or State-of-art.
Pushed content	Information pushed onto a recipients computer without the user directly requesting it.	Irrelevant, Minimal, Good practice, or State-of-art.
Personalization of interaction	Automatic selection of content or functionality based on a user's actual or perceived needs. (Used as a marketing tool as well as for handling information overload.)	Irrelevant, Minimal, Good practice, or State-of-art.

4.2) Discovery

This includes the discovery of useful or interesting information not related to the primary purpose of the system. Covers other indirectly related topics. (Currently the most common intended uses of supplementary learning is in website design, where related indirect topics are intentionally included in site content in order to please some of the visitors.)

Feature	Summary	Simple scale
Learning about people	Provision of information about people related to current context.	Irrelevant, Occasional, Explicit, or Dominating.
Learning about events	Reports of upcoming and completed events loosely related to the current context.	Irrelevant, Occasional, Explicit, or Dominating.
Learning about organizations	Provision of information about organizations and social groups indirectly related to current context.	Irrelevant, Occasional, Explicit, or Dominating.
Learning about tools and processes	Provision of information about the tools and processes used to improve their work efficiency and the quality of their results.	Irrelevant, Occasional, Explicit, or Dominating.
Learning about science and technology	Provision of information about how and why things work.	Irrelevant, Occasional, Explicit, or Dominating.
Learning about the world	Provision of information about mankind, nature, science, our planet and outer space.	Irrelevant, Occasional, Explicit, or Dominating.

4.3) Novelty

Novelty involves the provision of something unusual or new. The extent of novelty can be influenced by the user's familiarity with the domain. (N.B. High novelty frequently results in significantly reduced usability - see Usability measures and http://www.emmus.org/)

Feature	Summary	Simple scale
Personal experience	A system satisfying a user's previously unrecognized need or stimulus.	Irrelevant, Average, Unusual, or Exceptional.
Paradox/riddle	Provision of contradictory features or phrases to which a solution is promised.	Irrelevant, Average, Unusual, or Exceptional.
Dramatic experience	Provision of a novel state or situation in the traditions of story-telling, art or music making.	Irrelevant, Average, Unusual, or Exceptional.
Technical gimmicks	Provision of attention-getting device.	Irrelevant, Average, Unusual, or Exceptional.

4.4) Involvement

Emotional involvement includes the excitement or strong feeling, encouraging psychological (and physiological) effect. This can be addictive, encouraging return visits. Emotional involvement is used extensively in advertising, entertainment and news services.

In the context of a website, only some emotional features are desirable, while some others may be highly undesirable. In highly utilitarian websites, it is normal (and easy) to avoid all emotional involvement.

Feature	**Summary**	**Simple scale**
Confidence	Faith and trust in the future. Courage to handle events. (The inverse is the condition of "learned helplessness", in which all actions are believed to result in failure, despite new evidence to the contrary.)	Irrelevant, Occasional, Explicit, or Dominating.
Love	Strong affection for a person or group of people, based on admiration, benevolence, common interests or personal ties.	Irrelevant, Occasional, Explicit, or Dominating.
Pity	Empathy with one or more people in a significantly worse situation than the user.	Irrelevant, Occasional, Explicit, or Dominating.
Greed	Desire for acquisition, consumption or power.	Irrelevant, Occasional, Explicit, or Dominating.
Anger	Strong displeasure resulting from actual or perceived injury. Anger is often accompanied by a desire for action. The quality feature needs to be carefully combined with appropriate other features.	Irrelevant, Occasional, Explicit, or Dominating.
Fear	Anticipation or awareness of danger to self, or people with one has an affinity.	Irrelevant, Occasional, Explicit, or Dominating.
Lust	Strong carnal desire, devoid from love. This may have an immediate effect (positive or negative) that can greatly diminish the effect of most other quality features.	Irrelevant, Occasional, Explicit, or Dominating.
Selfishness	Disregard for others. Encouraging selfishness, and can aggravate racism and bigotry.	Irrelevant, Occasional, Explicit, or Dominating.

4.5) Comfort

Comfort includes the provision of reassurance and pleasure. Comfort is used to balance the difficulties typically associated with other rewards (novelty, supplementary learning, participation and emotional involvement). Some of the comfort features listed below are important only in some for certain types of product.

There is a superficial similarity between the System Reward of "comfort" and the system constraint of "usability". However comfort is a positive effect, and some people require as much comfort as possible. In contrast, usability involves only the reduction of pain in using the system, and it is only necessary to have "just enough" usability within the system.

Feature	Summary	Simple scale
Visual beauty	Visual impression designed to create positive emotion.	Irrelevant, Average, High, or Extreme.
Poetic act	Elegance of act or thought, with altruistic motive. Written poetry is one of many forms of poetic action.	Irrelevant, Average, High, or Extreme.
Engineering elegance	Provision of efficient structure with functional simplicity.	Irrelevant, Average, High, or Extreme.
Expectation	Use of themes that promise a later reward.	Irrelevant, Average, High, or Extreme.
Conformance to social norms	Adoption of political correctness and good morals.	Irrelevant, Average, High, or Extreme.

5) Conclusion

Web-site owners and developers dream of creating killer websites. The probability of achieving this can be greatly increased by identifying the priorities for System Rewards, then methodically improving the most important ones.

This paper is a start. There is still much research that is needed to improve the dichotomy of System Rewards (as presented in section 4), to provide suitable measures and to provide appropriate quality-improvement schemes. However the key lesson is that engineering methods can be applied to support creative and commercial activities.

References

Boonstra, F., "Definition of external metrics", in *MultiSpace Quality Pack 2: Quality specification: questionnaires and metrics*, EP23066-D3.2C (Feb 3, 1998).

Carter, D. E. and Baker, B. S. *Concurrent Engineering: The Product Development Environment for the 1990's.* Reading, MA: Addison-Wesley, 1991, ASIN 0201563495.

Cowderoy, A.J.C., Daily, K., *The MultiSpace Framework*, ESPRIT EP23066, 1997. Available at http://www.mmhq.co.uk/multispace/d2-1p.pdf

Cowderoy, A.J.C., Donaldson, A.J.M., Jenkins. J.O., "A metrics framework for multimedia creation", *Proc. of the 5th International Software Metrics Symposium*, Nov 20-21, 1998, Maryland, USA, IEEE Computer Society, ISBN 0-8186-9201-4.

Cowderoy, A.J.C., "Measures of size and complexity for web-site content", ESCOM SCOPE 2000, April 18-20, 2000. Published in *Project Control, the Human Factor*, Shaker. Available at http://www.mmhq.co.uk/papers/escomscope2000.pdf

Fellenstein, C., and Wood, R., *Exploring E-Commerce, Global E-Business and E-Society,* Prentice Hall, 2000, ISBN: 0130848468.

Siegel, D., Futurize Your Enterprise: Business Strategy in the Age of the E-customer, John Wiley & Sons, 1999, ISBN: 0471357634. Website: http://www.futurizenow.com/

Williamson, O.E., *Markets and Hierarchies: Analysis and Antitrust Implications,* Free Press, 1983, ISBN 0029347807.

van Veenendaal, E.P.W.M., "Quality characteristics for multimedia systems", Ch.3 of *The MultiSpace Priorities,* EP23066-D2.2P, 1998. Available at http://www.mmhq.co.uk/multispace/d2-2p.pdf

QWE2000 Session 9M

Dr. Esther Pearson [USA]
(Genuity Corporation)

"Website Operational Acceptance Testing; Process Assessment and Improvement"

Key Points

- Design of an Website Operational Acceptance Test Environment
- Building Relations with Functional Test and Development Organizations
- Website Test Strategies and Tactical Planning
- Performing Website Operational Acceptance Testing

Presentation Abstract

It is customer expectations that a website would remain up and running on a 24x7and 365 days per year basis. The Internet does not open at 9am and shutdown at 5pm Monday through Friday and stay closed on weekends. It is used as a tool for buying and selling of goods and services, research of information, communication and entertainment every day and every hour.

With this in mind, one of the most critical aspects of testing a website before it is deployed live is website "Operational Acceptance Testing". This testing provide a high-level of assurance that websites can be provisioned and deployed live in a systematic manner. This systemic process ensures efficient and effective provisioning of one or a thousand servers and greater probability of 24x7 and 365 days up time of customer servers and web presence once deployed live.

This presentation will explore the test efforts needed by Internet Service Providers (ISP's). The efforts must include not only functionality testing of software integration but include Operational Acceptance Testing.

About the Speaker

Dr. Esther Pearson is the Director of Quality Assurance and Test for Genuity Corporation. In her role as director, she manages a group of engineers which test web applications in an Internet web hosting environment. Dr. Pearson has worked in the computer, network and Internet industries for over 20 years.

GENUITY

Website Operational Acceptance Testing: Process Assessment and Improvement

Dr. Esther Pearson
Director

QWE2000 -Brussels, Belgium

GENUITY

Outline

- Introduction
- Design of Website Operational Acceptance Test Environment
- Building Relations with Functional Test and Development Organizations
- Website Test Strategies
- Tactical Planning
- Performance of Website Operational Acceptance Testing
- Test Process Assessment and Improvement

GENUITY

Introduction

Dr. Esther Pearson

- Director Production Acceptance Services
- 25 years Experience in QA/Test and Engineering
- Developed Production Environment Acceptance Test Process for Genuity

GENUITY

Introduction

GENUiTY

- Genuity (NASDAQ: GENU) is a leading e-business network provider of high-quality, managed Internet infrastructure services to enterprises and service providers. Genuity offers a comprehensive suite of managed Internet infrastructure services to deploy corporate intranets, and business-to-business and business-to-consumer extranets, including: Internet access through dial-up, dedicated and digital subscriber lines; Web hosting and content delivery; and value-added e-business services such as Virtual Private Networks (VPNs), security services and Voice-over-Internet Protocol (VoIP). Based in Burlington, Mass., Genuity has offices and local partnerships throughout the U.S., Europe, Asia and Latin America.

GENUITY

Introduction

Website Operational Acceptance Testing

- Testing of a Website presence in a Production Environment to ensure site setup and configuration meets quality standards prior to site “live” deployment.

GENUITY

Introduction

Test Process Assessment and Improvement

- Questions to be Answered…
- What is Operational Acceptance Testing?
- How is a Test Environment Designed?
- What Organizational Relationships are necessary for Operational Acceptance Testing?
- How is Operational Acceptance Testing Performed?
- How is Test Process Assessment and Improvement Implemented?

GENUITY

Design of Test Environment

Operational Acceptance Test Environment

- ISP's
- Data Centers
- Acceptance Test Process

GENUITY

Design of Test Environment

ISP's (Internet Service Providers)

- Provide Website presence on the Internet

Data Centers

- Provide secure facility for Website Servers and Equipment housing

GENUITY

Design of Test Environment

Acceptance Testing Process

- Ensure Site is properly Setup and Configured
- Final System Test Prior to site “live” deployment

GENUITY

Building Relations

Building Relations with Functional Test and Development Organizations

- Key Areas of Operational Test Environment
- Production Environment
- Test Racks for Production Equipment/Servers
- Production Equipment/Servers and Software

GENUITY

Website Test and Planning

Website Test Strategies and Tactical Planning

- Test Process
- Test Methods
- Tactical Implementation

GENUITY

Performing Testing

Performing Website Operational Acceptance Testing

- Website Testing
- Operational Testing
- Acceptance Testing

GENUITY

Performing Testing

Acceptance Testing

- Certification of Software
- Certification of Systems
- Acceptance into “live” Production

GENUITY

Summary

Test Process Assessment

How is Test Process Assessment Implemented?

- Final System Test implemented to detect any previous Process, Setup, Provisioning and Configuration Problems

GENUITY

Summary

Test Improvement

How is Test Improvement Implemented?

- Problems detected in Final System Test must be reviewed and fed back into the overall site deployment process to detect and prevent problems reaching the final acceptance test effort.
- Continuous improvement within the entire test process and organization involved in the deployment process is achieved.

WEBSITE OPERATIONAL ACCEPTANCE TESTING: PROCESS ASSESSMENT AND IMPROVEMENT

By:
Dr. Esther Pearson

Fourth International Software Quality Week Europe (QWE2000)
Brussels, Belgium
20-24 November 2000

Abstract

It is customer expectations that a website would remain up and running on a 24x7and 365 days per year basis. The Internet does not open at 9am and shutdown at 5pm Monday through Friday and stay closed on weekends. It is used as a tool for buying and selling of goods and services, research of information, communication and entertainment every day and every hour.

With this in mind, one of the most critical aspects of testing a website before it is deployed live is website "Operational Acceptance Testing". This testing provide a high-level of assurance that websites can be provisioned and deployed live in a systematic manner. This systemic process ensures efficient and effective provisioning of one or a thousand servers and greater probability of 24x7 and 365 days up time of customer servers and web presence once deployed live.

This presentation will explore the test efforts needed by Internet Service Providers (ISP's). The efforts must include not only functionality testing of software integration but include Operational Acceptance Testing.

About the Speaker

Dr. Esther Pearson is the Director of Production Acceptance Services, a Quality Assurance and Test organization at Genuity Corporation. In her role as director, she manages a group of engineers who test web applications and site configurations in an Internet web-hosting environment. Dr. Pearson has worked in the computer, network and Internet industries for over 20 years.

About GENUiTY, Inc.

Genuity is a leading e-business network provider of high-quality, managed Internet infrastructure services to enterprises and service providers. Genuity offers a comprehensive suite of managed Internet infrastructure services to deploy corporate intranets, and business-to-business and business-to-consumer extranets, including: Internet access through dial-up, dedicated and digital subscriber lines; Web hosting and content delivery; and value-added e-business services such as Virtual Private Networks (VPNs), security services and Voice-over-Internet Protocol (VoIP). With extensive IP experience, the company integrates its suite of services into corporate networks and delivers high-performance, secure and scalable infrastructure services for conducting business on the Internet. Based in Burlington, Mass., Genuity has offices and local partnerships throughout the U.S., Europe, Asia and Latin America.

Table of Contents

Introduction

This document introduces the process and activities to be performed in the design, development, implementation and assessment of Website Operational Acceptance Testing. The purpose of this paper is fivefold:

1. It details the design of a Website Operational Acceptance Environment

2. It describes building relations with functional Test and Development organizations to ensure collaborative quality efforts.

3. It provides Website test strategies for an operations production environment.

4. It provides tactical planning and performance of Operational Acceptance Testing.

5. It can be used to create a structure for test process assessment and improvement.

The first section of this paper defines the Operational Acceptance Testing nomenclature. The environment in which Acceptance Testing is conducted. The environment design which includes the test equipment, network and systems.

The second section depicts the organizational relations, strategies, planning and performance of the test effort. Some of these are standard practices of test and quality with the point of differentiation being the environment in which the test effort is occurring.

The final section of this document highlights the assessment of the test effort, as well as, on-going improvements of the test effort. These improvements to ensure, as always, continuous improvement of testing and product under test.

Design of Website Operational Acceptance Test Environment

Operational Acceptance Testing is the combination of a number of test strategies and methods. It is a combination because it views the test effort as “systems” test, which takes into account hardware, software and network considerations. With this in mind, definitions must be given to ensure a common language when describing testing.

The Acceptance Test effort is comprised of several areas. These areas can be defined as follows.

Functional Acceptance Test
Testing performed to ensure that hardware and software installed and configured in a website environment is functioning correctly. The items include: (1) the network operating software and applications are properly installed and configured for access and operability; (2) the servers are installed and configured for connectivity; and (3) network hardware including routers and switches are installed and configured for network and internetwork operations.

Site Acceptance Test
Testing to ensure the Firewall and VPN configurations are properly setup, configured and provides accessibility and security. The Firewall testing consists of verifying the firewall policy and required customization to meet customer requirements. The VPN testing consists of ensuring an encrypted tunnel with requested authentication are in place.

System Acceptance Test
Testing to ensure proper functioning of the site configuration, which includes software, hardware, monitoring, backup, network, firewall, security and high availability operations within a hosting environment. Also, auditing of the support systems to ensure accurate and complete data exist to profile site configuration and the customer identification in case of customer problem escalations.

Business Solution Acceptance Test
Testing to ensure the customers purpose for the site or the customers business strategy is being exercised and tested for valid operation. This test utilizes customer site utilization scenarios, which are also known as, user interface and performance tests.

Building Relations with Functional Test and Development Organizations

Operational Acceptance Testing is the final test effort prior to a website going "live" on the Internet. Operational Acceptance Testing is a "system" level test which occurs after unit or component-level development testing occurs by the development engineers or their assigned functional engineering-based QA organization.

This testing ensures that the loaded and configured units of software, applications and hardware that were developed in-house or by a third-party vendor are integrated and functioning properly. This testing does not look at the website as a system, but as individual components. Wherein the individual components are tested for proper functionality eliminating some of the variables that could affect the ability to obtain accurate results from Operational Acceptance Testing.

To ensure component testing is performed effectively it is important to maintain a collaborative relationship with the component testers and the development organization. This relationship provides a venue to review component level test results and fixes that are incorporated by the development organization. As well as, other synergies that are gained through utilizing the experiences gained during component level testing.

Thus, building relations with the functional test and development organizations reaps rewards of knowing component-level testing has occurred and the results of that testing. This knowledge shapes the Operational Acceptance Test efforts.

Website Test Strategies

Testing of a website requires strategies that incorporate the Data Center production environment. In other words testing is conducted in a “live” environment, but the site is not yet turned over to the customer.

With this in mind, it is required that the website as a system and as a business solution, is exercised. This is done to determine if the website will function as requested by the customer. But not only as it is requested to function by the customer, but as it is needed to operate by the Internet Service Provider (ISP). The Internet Service Provider needs to be able to monitor the activities, events and critical running processes of the website and also backup/restore any critical stores of data.

These two requirements are the key strategies for Operational Acceptance Testing. Stated another way, the key strategies for website testing are to ensure the website functions as requested by the customer and as needed for supportability by the ISP. A core set of Operational Acceptance Test categories is as follows:

- Operational Support Tests
- Website Server Tests
- Application Tests
- Database Tests
- Firewall Tests
- VPN Tests
- Failover Tests
- Redundancy Tests

Tactical Planning

To carryout the categories of Operational Acceptance Tests tactical planning is needed. This planning consists of more than simply providing testbeds and personnel. It consists of creating an environment, which replicates the environment in which the site will exist.

Creating this environment requires placing testbeds within the web-hosting organizations Data Center. Thus, testing occurs in the exact environment in which the site will be deployed, accessed and utilized. This may appear unusual but is necessary to provide an accurate testing environment.

The planning of tests that will be run to exercise the website under test consists of two types. These are support and functional tests. The support tests ensure that the ISP will be able to provide technical support of the website. The functional tests ensure the website's components are integrated and running as a system.

The tactical plan for "support" consists of a validation and audit of the web servers and network devices. Validation checks the servers and network devices are setup and configured correctly for access by the ISP and outside customers. For example, the IP addresses for all servers and network devices would be validated and an audit would be made to ensure these IP addresses were documented in support databases in case their was a need to ping the server or other devices to ensure their connectivity.

The tactical plan for the "functional" tests consists of exercising and verifying that the website's servers are integrated with each other and functioning to share processes and data. For example, if the website has a database server; that server can be populated with data from user interactions; the database server can then be queried to obtain information; and the database server can also be accessed and information copied onto a backup server. Thus, each component of the website is functioning as an integrated system.

Therefore, with these tactical testing needs in mind, test scenarios are developed with the intent of ensuring technical support can occur and the website functions as a system.

Performance of Website Operational Acceptance Testing

Performance of Operational Acceptance Testing ensures the website components function together as a system. This is the actual exercising of the system or “running” the test scenarios to verify the website as a system. To perform this verification it is key to perform testing in a “production” environment also known as a Data Center environment.

The testing consists of the standard processes of test, report detected problems, and re-test to verify detected problems are fixed. The problems that are detected are used a information points to improve on-going website setup and configurations. An example of an Operational Acceptance Test is summarized in the Appendix.

Once the testing is completed and problems fixed, the website is ready to go “live” in a production environment, wherein the website is turned over to the customer. This turn over to the customer does two things it certifies the site as a functioning system and it places the site into a “support” mode.

The support mode indicates if any problems occur on the site or the need arises for upgrades or changes they will be handled as a customer escalation or site maintenance need.

Test Process Assessment and Improvement

The Test Process Assessment and Improvement is a phase of analyzing the problems detected during testing. This analysis is meant to uncover root causes. These root causes will then be investigated to determine if they are the result of:

- Design Defects
- Provisioning (setup and configuration) Errors
- Hardware, Software, Application or Network Faults
- Personnel Training Needs

Once the root causes are identified they are addressed to ensure improvement of processes occur and root problems are eliminated. Thus, methods of continuous process improvement are put in place. These methods benefiting on-going website operations.

All organizations that are responsible for the root problems receive communication about the problems, thus providing opportunities for correction and continuous improvement on an organizational, as well as, process level.

Appendix

Operational Acceptance Test Summary

No:	System Solutions Test	Test Scenario Overview
	Operations Tests	Ensure supportability of Customer Site
3	Hardware Test	Verify customer hardware inventory
4	Hardware Configuration Test	Verify hardware configurations
5	Software Test	Verify customer software inventory
6	Software Configuration Test	Verify software configurations
7	SystemPlus Verification	Verify Customer Support Data
8	Magma Plus Verification	Verify Customer Profile Information
9	Terminal Server Access Test	Confirm Remote access to system consoles
10	Root Account Password	Verify Root Account Changed for customer
11	Power Cycling Test	Confirm Systems Remotely Power Cycled
12	Site Connectivity Test	Verify site can ping Genuity system
13	Backup Verification	Verify System Backup & Restore
14	Monitoring Test	Verify System Monitors for URLs & Processes
15	Account Verification	Confirm Customer Accounts Setup
16	Telnet Test	Verify Telnet to Systems
17	FTP Test	Verify Accounts can FTP to System
18	SSH Test	Verify "ssh" to systems
19	Login Test	Verify SecureID
20	Nameserver Test	Verify Nameserver entries
21	/etc File Test	Verify /etc configuration setups
22	Hosts.allow.deny Test	Verify host.allow and host.deny configurations
23	Ftpd/ftpaccess File Test	Verify ftpaccess file configuration
	System Validation Test	**Validate Software, Hardware and Server systems**
25	Application Test	Verify Real Media, Verisign Certificate, ect…
26	Raid Test	Verify Raid setup
27	Webserver Test	Verify http and https services
28	Database Test	Verify ODBC backend
	VPN Test	**Verify VPN Configuration and Connection**
30	OOB Test	Verify Out of Band connection
31	Modem Test	Verify modem connection
32	Traceroute Test	Confirm Trace
	Site Patrol Test	**Verify Firewall Management and Security Policy**
34	Firewall Test	Verify port connections
	Site Replicator Test	**Verify Sync, Replication info and Logs**
36	Configuration Test	Verify configuration, scheduler, content replication
37	Log Rotation Test	Verify Log successfully rotated
	Distributed Hosting Test	**Verify Director, Redirector and Balancer systems**
38	Load Balancer / Local Director Test	Verify Load Balancer configuration Verify Traffic Redirection configuration
39	Distributed Director Test	Verify Distributed Director configuration Verify Failover and Redundancy

QWE2000 Vendor Technical Presentation VT 11

Bruno Bouyssounouse (PolySpace)

" Detect All Run-Time Error Without Test-Beds"

Key Points

Key points to be supplied.

Presentation Abstract

PolySpace Verifier checks C applications exhaustively and automatically, for all possible executions, without running the software.

- Exhaustive, automatic detection of run-time errors

- Reduce validation costs dramatically (no test-beds)

- No changes required to existing processes

- No constraints on coding style

About the Speaker

To be supplied.

QWE2000 Vendor Technical Presentation VT10

Edward Miller
(eValid)

The Argument for Client-Side Testing

Key Points

The Web is a complex place. There is much that is very importnat that can go wrong. What really counts in terms of quality is how users perceive a site. Because the client view is all-important, eValid's approach to WebSite testing is through a Test Enabled Web Browser.

This presentation outlines the reasons why the client-side view is important, and describes how a Test Enabled Web Browser can help sort out the WebSite quality issue.

About the Speaker

Dr. Edward Miller is President of Software Research, Inc., San Francisco, California, where he has been involved with software test tools development and software engineering quality questions. Dr. Miller has worked in the software quality management field for 25 years in a variety of capacities, and has been involved in the development of families of automated software and analysis support tools.

He was chairman of the 1985 1st International Conference on Computer Workstations, and has participated in IEEE conference organizing activities for many years. He is the author of Software Testing and Validation Techniques, an IEEE Computer Society Press tutorial text. Dr. Miller received his Ph.D. (Electrical Engineering) degree from the University of Maryland, an M.S. (Applied Mathematics) degree from the University of Colorado, and a BSEE from Iowa State University.

The Argument for Client-Side Testing

Why WebSite Testing Is Best Performed
From The Same Perspective As A User:
With A Browser

eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA

Phone: 1-415-550-3020
FAX: 1-415-550-3030
Web: www.e-valid.com
Email: info@soft.com

Most Common Problems in WebSites

- Quality/Content
  - Broken Links
  - Missing Components
- Performance
  - Too-Slow Download
- Interaction
  - Failed 1st Layer Transactions
    - Login
    - Specialized Controls
  - Delayed 2nd Tier Transactions
  - Delayed 3rd Tier Transactions

Alternative Technologies

- Windows
  - Client/Server Testing
  - Windows Events
  - Browser is "opaque"
- Unix
  - Client/Server Testing
  - X-Display Events
  - Browser is "opaque"
- Browser
  - Everything is open

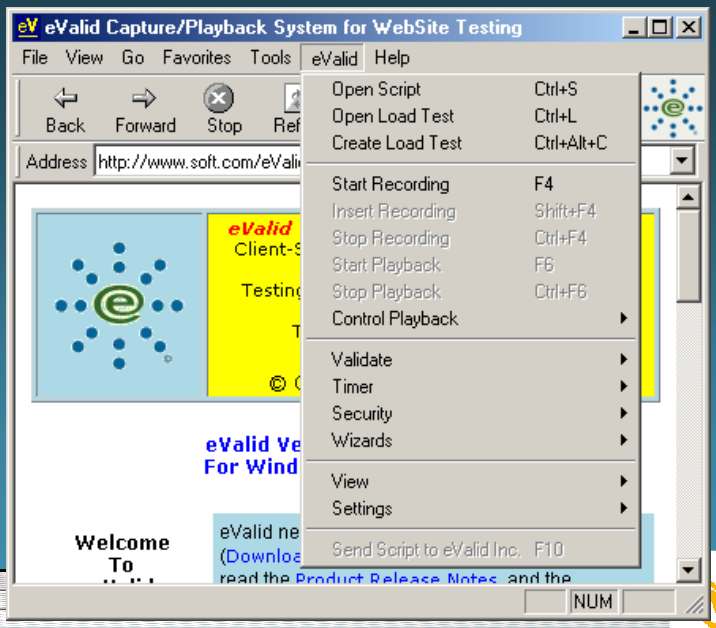
WebBrowser Testing Pros/Cons

- Pros
  - 100% User View
  - Realistic
  - Natural operation
  - Accurate timings
- Cons
  - Not all browsers are alike
  - UNIX platform support limited

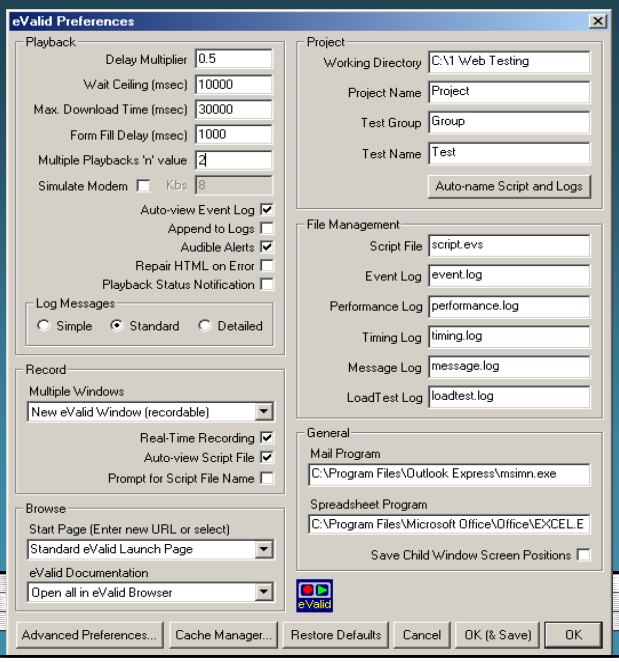
eValid General Features

- IE Base
- Simple Script Language
- Point and Click Interface
- Online Documentation
- Advanced Recording
- Variety of User Options

eValid GUI with Pulldowns



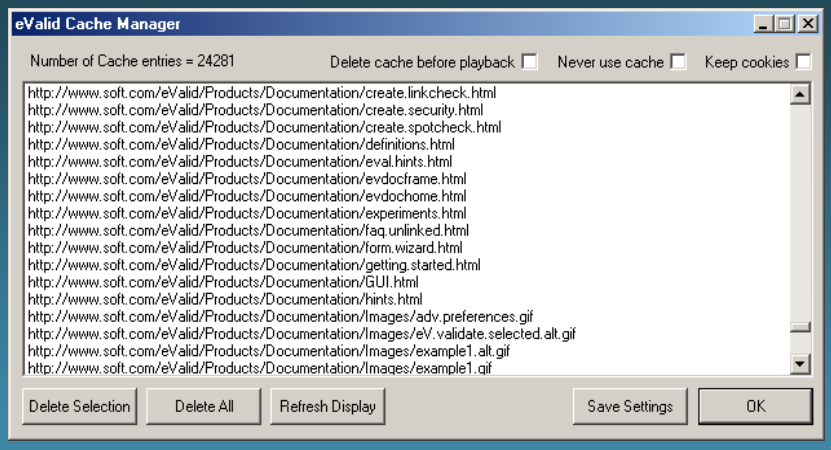
eValid GUI with Preferences



eValid Cache Management

- With Cache
- Clear Cache Before Execution
  Realistic
- No Cache
- Cookies

eValid GUI with Cache Management



Advanced Recording Capabilities

- Java Applets
- ActiveX Controls
- Scrolling
- Sub-Browsers
- Secure Sites/Logins

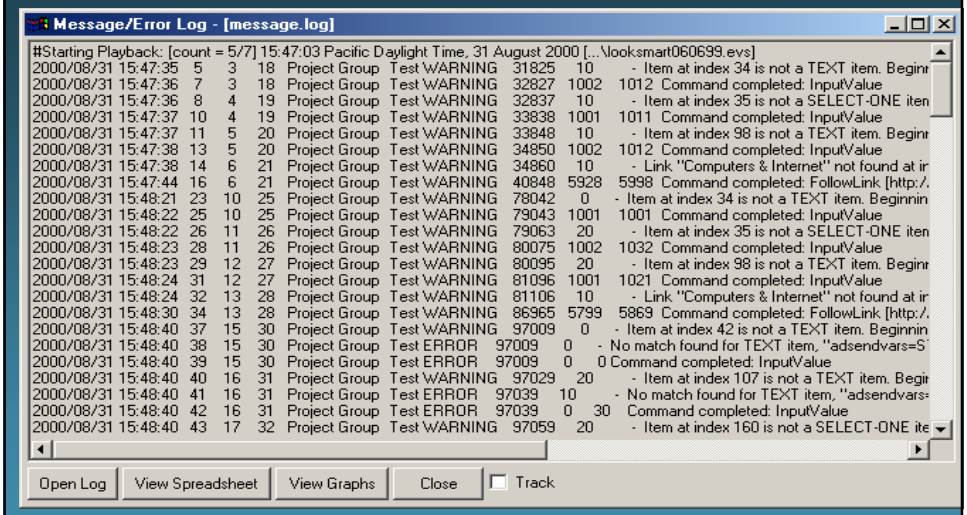
eValid Static Testing/Monitoring

- Wizards
  - Links
  - Forms
  - Buttons
- Timing
- Image Capture

eValid Functional Testing

- Record/play
- Validation
- Content
- Images
- Links

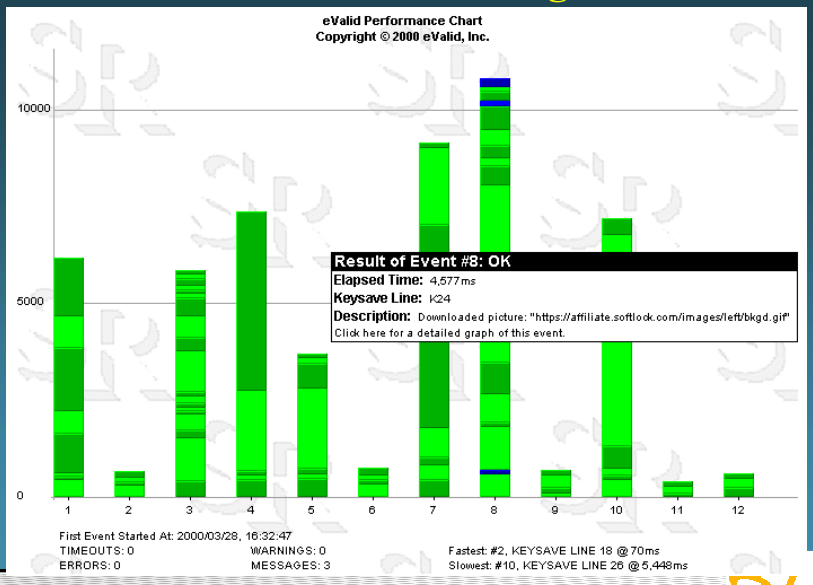
eValid GUI with Error Log Report



eValid Performance Testing

- Detailed Timings
  - DNS
  - 1st Byte
  - Base Page Download
  - Component Downloads
  - Page Rendering

eValid GUI with Tuning Chart



Loading Using HTTP "Pinging"

- Pros
  - Economical (Free!)
  - Widely Available
  - Easy
  - E.g. "TorturePL"
- Cons
  - Unrealistic
  - Possible Self-Deception
  - Possible False Readings

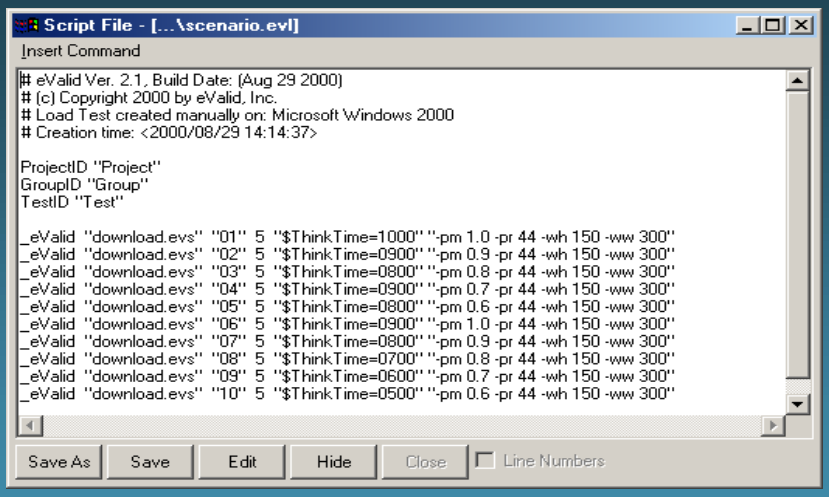
Load Testing Using Browser

- Pros
  - Realistic
  - Repeatable
  - Scalable Scenario
  - Simple Reports
- Cons
  - Bandwidth Requirement
  - Programming

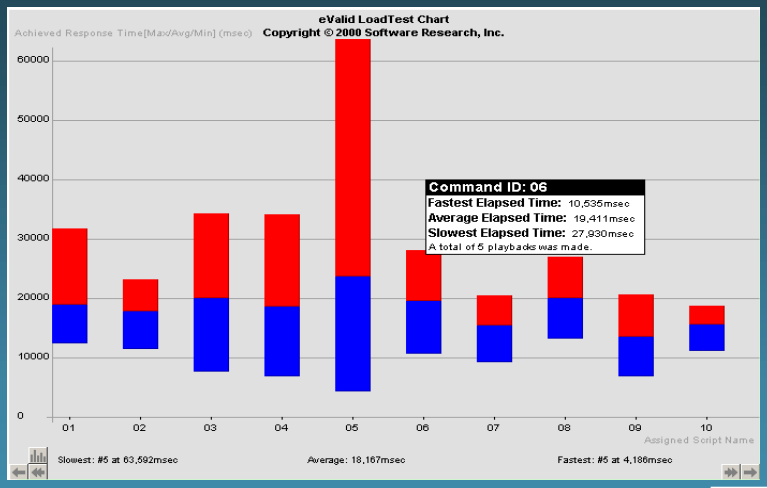
eValid Load Testing

- Scenario Composition
- Realistic Users
- User Descriptions
- Cache Management
- Reporting

eValid GUI with Scenario Script



eValid GUI with Scenario Timing Chart



Conclusions
&
Recommendations

QWE2000 Session 10T

Dr. Ray Paul(OASD) & Dr. Wei-Tek Tsai(Arizona State University) [USA]

"Assurance-Based Testing: A New Quality Assurance Technique"

Key Points

- Statistical assurance
- Testing processes
- Experience in statistical assurance

Presentation Abstract

This paper presents statistical assurance techniques for testing. Even though it has been developed to Y2K testing, it can be applied potentially any software testing project. The technique has been developed by the Department of Defense (DoD) and used in a variety of projects within DoD.

The ABT test process and the underlying statistical models provide DoD managers with objective test data and a specific quantitative level of confidence in the test results. Furthermore, the ABT process can be embedded within the existing DoD Y2K testing processes to minimize changes to the existing testing practices. The statistical models explicitly model Y2K faults, address regression testing of Y2K modifications, and provide specific and quantitative output. Also, the ABT process can be used for many future test programs, including both commercial and government applications.

About the Speaker

Ray Paul is current the Directorate at DoD OASD Y2K Office. He is in charge of DoDËs recent Y2K testing effort. He recently received his Ph.D. in Computer Science and Engineering.

W. T. Tsai is currently Professor of Computer Science and Engineering at Arizona State University. He received his Ph.D. and M.S. in Computer Science from University of California at Berkeley. He has been involved in software engineering research.

Assurance-Based Testing: A New Quality Assurance Technique

Ray Paul
OASD C3I (I&A)

W. T. Tsai
Arizona State University

Assurance Based Testing

- Originally developed for Y2K, but it can be used in any software testing projects.
- Y2K testing requires objective and quantifiable data to ensure confidence.
- Need a decision making tool to determine the quality of test done.

Goal

- A process that can be embedded within the current DoD testing process.
- Minimal changes to the current testing process.
- Easy to use.
- Minimize extra effort needed to perform the ABT.
- Encourage reuse of existing test resources and results.
- Use the minimum number of test cases while maintaining the desired quality.

The ABT Processes

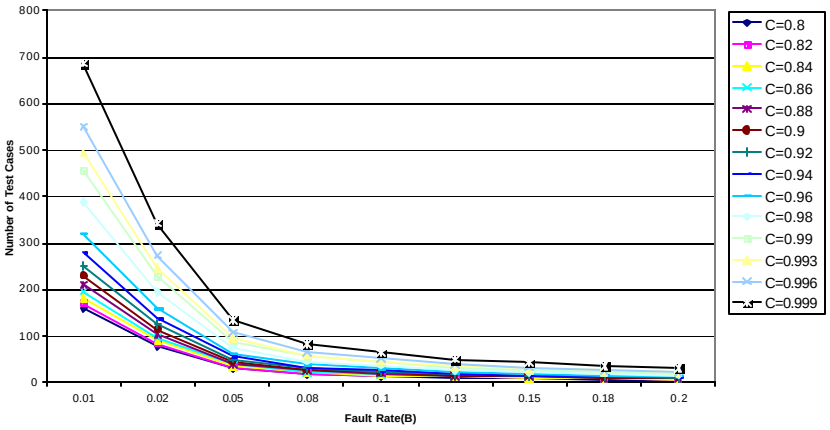
- Top-down process
  - Starts from the system testing, then to integration testing and finally to module testing if necessary.
  - The process will stop whenever the system passes the ABT requirements at that level.
  - This is the process to use whenever the testing is almost done.
- Bottom-up process
  - Can be embedded within the existing testing process.
  - Starts from module testing, to integration testing, and finally to system testing.
  - If the process stops when the system passes the ABT requirements .
  - This is the process to use if we wish to use the ABT during the system development and test processes.

ABT Statistical Model (I)

- $1 - (1-B)^N = C$
  - C is the confidence level desired.
  - B is the failure density(threshold).
  - N is the number of test cases required.
  - N = Ln(1-C)/Ln(1-B)

Confidence Level(C)	Failure Density(B)	Number of test cases required
0.8	0.01	160
0.8	0.02	80
0.8	0.05	31
0.8	0.10	19
0.95	0.01	298
0.95	0.02	148
0.95	0.05	58
0.95	0.10	28

ABT Statistical Model (II)

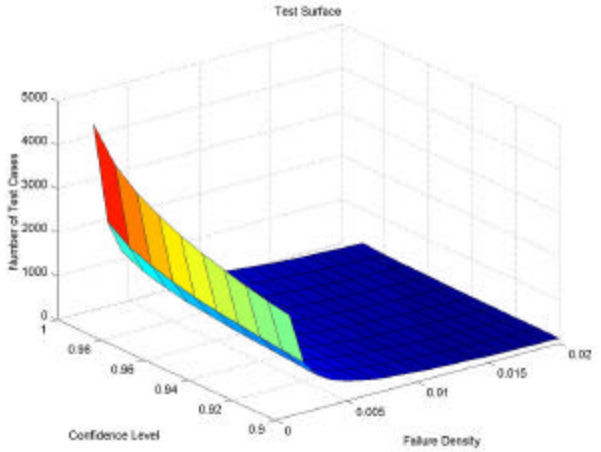


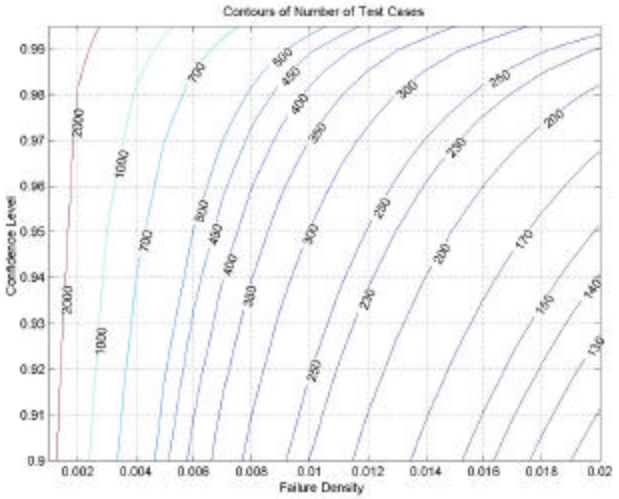
Determine the Confidence Level

- Determine the confidence level required for a given system before conducting the ABT process.
- For mission-critical system this can be 0.95, otherwise a lower confidence level, such as 0.8.
- The higher the confidence level, the more test cases are required.
- Conflict between the desired confidence level vs. the extra effort needed.

Determine the Target Failure Rate

- Determine the failure rate from the system requirements.
- If the system is mission-critical and safety-critical, use a low failure rate.
- The lower the failure rate, the more the number of test cases needed.
- The following figures show the number of test cases increase rapidly when we have low failure rates and high confidence.


Test Surface
Number of Test Cases
5000
4000
3000
2000
1000
0
1
0.98
0.96
0.94
0.92
0.9
Confidence Level
0
0.005
0.01
0.015
0.02
Failure Density

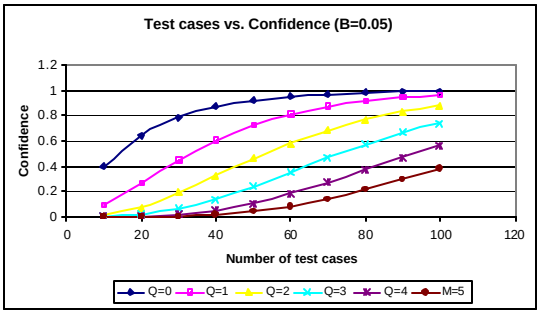

Contours of Number of Test Cases
Confidence Level
Failure Density
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.9
0.002
0.004
0.006
0.008
0.01
0.012
0.014
0.016
0.018
0.02
2000
1000
700
500
450
400
350
300
250
230
200
170
150
140
139

Problems with the Original Model

- The original model requires the ABT process to be repeated whenever a single failure is encountered.
  - The software is modified and then another round of the ABT is applied.
  - This is simply too expensive.
  - Practitioners do not have time and energy to do the ABT even twice.

The Modified Model

- Include failure case in the model,
- Whenever there is no failure, the extended model becomes the original model.
- As the failure increases, the confidence decreases rapidly.
- As the target failure density decreases, more test cases or fewer failures are required to achieved the same confidence.



The ABT Process based on the Modified Model

- Similar to the previous processes, but now failures are allowed. Thus we need to apply the ABT process only once at each level.
- It is this process that is used in all DoD exercises.
- The modified model can be further extended to include ripples and specific testing features such as Y2k failures.

The 3-Step Process

- Step 1: Run the regression testing. If the system fails at this step, it should be rejected; otherwise go to the next step.
  - If the modified system cannot pass this step, the statistical models say that it is highly unlikely that the system will be able to pass the ABT requirements.
  - This step is relatively cheap because it reuses the existing test resources only.
- Step 2: Run Y2K specific test cases. If the system fails at this step, it should be rejected; otherwise go to the next step.
  - If the modified system cannot pass this step, it will not be able to pass the next step.
  - This step is cheaper than the next step.
- Step 3: Run test cases for Y2K ripples. Reject the software if it does not pass the ABT requirements.
  - This step is more expensive because it involves ripple effect analysis.

Benefits and Experience of ABT

- The ABT top-down process has been used at various DoD sites.
- The feedbacks are positive and in many cases this is the first time they reported any objective and quantifiable data to decision makers and test engineers.
- The ABT exercise is done with assistance of the ABT website where statistical models are calculated and visually displayed.

Benefits and Experience of ABT (II)

- It is relatively easy to identify which parts of subsystems are over tested and which are under tested.
- The testing team indicated that they can easily incorporate the ABT requirements in their test projects if they were informed at the beginning of the project.
- The testing team indicated that it is easy to apply the ABT process after some training.

Assurance-Based Testing: A New Quality Assurance Technique

Ray Paul
OASD C3I (I&A)
Department of Defense
Washington, D.C.

W. T. Tsai
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287

1. Introduction

This paper presents statistical assurance techniques for testing. Even though it has been developed to Y2K testing [2], it can be applied potentially to any software-testing project. The technique has been developed by the Department of Defense (DoD) and used in a variety of projects within DoD.

The ABT test process and the underlying statistical models provide DoD managers with objective test data and a specific quantitative level of confidence in the test results. Furthermore, the ABT process can be embedded within the existing DoD Y2K testing processes to minimize changes to the existing testing practices. The statistical models explicitly model Y2K faults, address regression testing of Y2K modifications, and provide specific and quantitative output. Also, the ABT process can be used for many future test programs, including both commercial and government applications.

The ABT statistical model allows the overall number of test cases to be reduced while maintaining the quality of test with a specific confidence level. The ABT framework takes into consideration the quantity and type of regression testing that has been performed as part of a system renovation. ABT potentially can be used in any testing projects, at any phase of testing, where quality of testing done needs to be quantitatively assessed. It allows software managers and engineers to determine the amount of effort to meet the stated test confidence level, and if additional effort will be needed.

2. Characteristics of the ABT Process

The ABT process addresses three specific levels of test: module, end-to-end, and integration testing. The characteristics of the ABT process reduce the effort and resources required for testing, including:

a. Reuse of prior test resources and results;
b. Insertion of ABT practices in existing test procedures to minimize changes in the existing test program;
c. Testing to a calculated level of test coverage, focusing on critical areas and likely failure points;
d. Use of the minimum number of test cases to achieve a specific confidence level.

The ABT process may be conducted with either a bottom-up or top-down approach. The bottom-up process can be used during software development. The top-down process can be applied to projects where testing has been almost completed. Both of these processes start with determining the threshold failure density and desired confidence level. The threshold failure density is determined by the requirements for the number of daily transactions and the criticality of failures. For example, if the system is mission-critical and no failure can be tolerated, the threshold should be low, such as 0.0001 or one failure every 10,000 operating hours. Once the threshold failure density is determined, the confidence level can be determined, usually by system criticality requirements. Note that a higher confidence level and lower failure density threshold will increase the number of test cases needed.

This top-down process assumes that the software has already been thoroughly tested, including module, integration, and end-to-end testing. Top-down ABT starts at the end-to-end test level, and if the system achieved the confidence level with the given target failure rate the process ends. Otherwise, the process proceeds to the next decomposition level, and performs ABT at that level. The process terminates when the confidence level and target failure rate achieved, or will proceed to module testing.

The bottom-up process goes hand-in-hand with the existing testing process, i.e., starts from module testing, to integration testing, and finally to end-to-end testing. At each stage, if the software satisfies the stated confidence and target failure rate, the process goes to the next stage. Otherwise, the software is rejected and must subject to further testing before another subject to anther ABT.

3. The ABT Statistical Models

The ABT statistical models determine the specific number of tests needed to achieve a desired level of confidence in the results. The failure density of a program is defined as the probability that the program will fail on a random input. It can also be thought of as the ratio of the number of input points at which the software fails to the total size of the input space assuming a random sampling is used. The ABT is a process to establish an upper bound on the failure density with a quantifiable statistical guarantee that the actual failure rate will not exceed this upper bound.

The first statistical model the ABT uses is developed by Howden [1]:

$$C = 1 - (1 - B)^N ,$$

In this formula, C is the confidence level, B the threshold or target failure density, and N is the number of random test cases that must be run sequentially without failure. This formula says that that when N random and independent tests are run sequentially without a failure, one has a confidence C that the actual software failure density is no more than B.

In other words, this is a *hypothesis testing* approach. The approach says that if N test cases are successful, the hypothesis may be wrong, i.e., the actual failure rate is greater than B, with a

probability of at most *(1 - C)*. It says that if N test cases are successful, one is *C %* confident that the actual failure rate is less than *B*. If the software actually fails on some input, the fault is identified and corrected and the same hypothesis testing approach is used again.

Figure 1 and Table 1 show the number of test cases required for various *C* and *B* combinations. Figure 1 shows the surface of *N* as a function of *B* and *C*. It shows how the required *N* sharply increases as the desired *C* increases and *B* decreases. The contours of this surface in the *(B-C)* plane are plotted in Figure 2. Note that for *B=0.016* and *C=0.91*, *N=150*. However, for *B=0.005* and *C=0.99*, *N=918*, a six fold increase.

Whenever a failure is encountered, he original Howden model requires the engineers to start over the ABT process. This is unfortunately an expensive proposal and practitioners simply do not have time and energy to repeat the ABT process. Thus, the original model is extended to allow failures where:

$$C = 1 - F = 1 - \sum_{k=0}^{Q} \binom{N}{k} B^k (1-B)^{N-k}$$

$$= \sum_{k=Q+1}^{N} \binom{N}{k} B^k (1-B)^{N-k}$$

where

$$F = \sum_{k=0}^{Q} \binom{N}{k} B^k (1-B)^{N-k}$$

With *N* random test cases executed with *Q* failures, one has the confidence *C* that the true failure rate is no more than *B*. In other words, with a probability of at least *C*, one will see more than *Q* failures in *N* test cases when the failure density is more than *B*. Table 2 shows some computation results.

This model has the following characteristics:

1. Confidence value is between 0 and 1.
2. The maximum confidence from a given set of *N* test cases is obtained when there are no failures. By substituting *Q = 0*, one can obtain the original equation.
3. As the failure increases, the confidence decreases rapidly. When all test cases result in failures, the confidence is zero.
4. As the targeted failure density decreases, more test cases or fewer failures are required to achieve the same confidence.

Figure 3 shows how the confidence varies with the number of test cases for the target failure density 0.05, for the failures between 0 and 5. Note that as the failures increases, the confidence decreases rapidly which is evident from the graphs becoming closer to the x-axis. When the failures increases from 0 to 5 out of 100 test cases, the confidence drops from 0.99 to around 0.4.

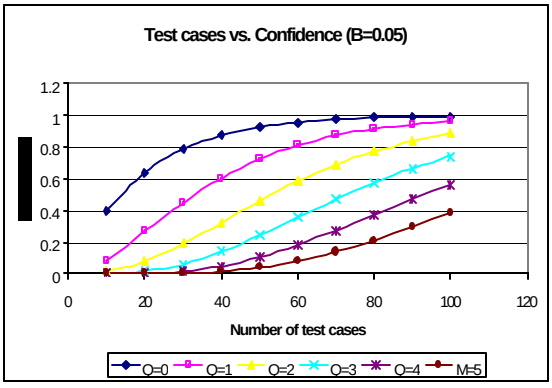


Figure 3 Confidence in the Presence of Failures

All the ABT exercises within DoD used the extended model because in practice failures are common.

The statistical model can also be extended to include analysis of ripples and specific faults such as Y2K faults [2]. Based on these models, customized processes can be developed to ensure specific aspects of various testing projects are carried out. For example, based on the ABT statistical models, the following three-step process can be used to ensure Y2K bugs are removed [2]:

Step (1) Run regression testing without considering the Y2K test cases. If the system fails to achieve the desired confidence level, it should be rejected. Otherwise;

Step (2): Develop specific Y2K test cases to test the software. If the system fails to achieve the desired confidence level, it should be rejected. Otherwise;

Step (3): Develop specific test cases to test Y2K ripples. If the system fails to achieve the desired confidence level, it should be rejected.

Based on the ABT statistical models, if the software could not even pass the regression testing without the Y2K test cases after the Y2K modifications, the software will surely fail the ABT requirements. Thus, if the software fails the first step, it should be rejected before carrying out the rest of the processes. Similarly, if the software does not pass the second step, it is not possible to pass the ABT requirements and should be rejected before carrying out the last step.

4. Benefits and Experience of ABT

The ABT top-down process has been used at several Y2K testing sites. The experience indicates that the ABT provides a reliable feedback mechanism for objective and quantifiable data to decision makers and test engineers. The ABT exercise was done with assistance of the ABT website where the engineers can use the ABT calculator and the ABT statistical models are visually displayed.

At one site, the ABT results demonstrated which parts of software were over-tested or under-tested. This kind of information is useful for both test planning and decision making. Furthermore, this is the first time these sites ever reported any objective and quantifiable data to assess the testing job performed. The testing team also indicated that they could easily incorporate the ABT requirements into the existing testing process had they been informed of the ABT at the beginning. They also indicated that it is easy to apply the ABT process.

5. References

[1] W. E. Howden, "Good Enough versus High Assurance Software Testing and Analysis Methods", in Proceedings of IEEE High Assurance Systems Engineering (HASE) Conference,1998, pp. 166-175.

[2] W. T. Tsai, R. Paul, W. Shao, S. Rayadurgam and J. Li, "Assurance-Based Y2K Testing", Proceedings of IEEE HASE, 1999.

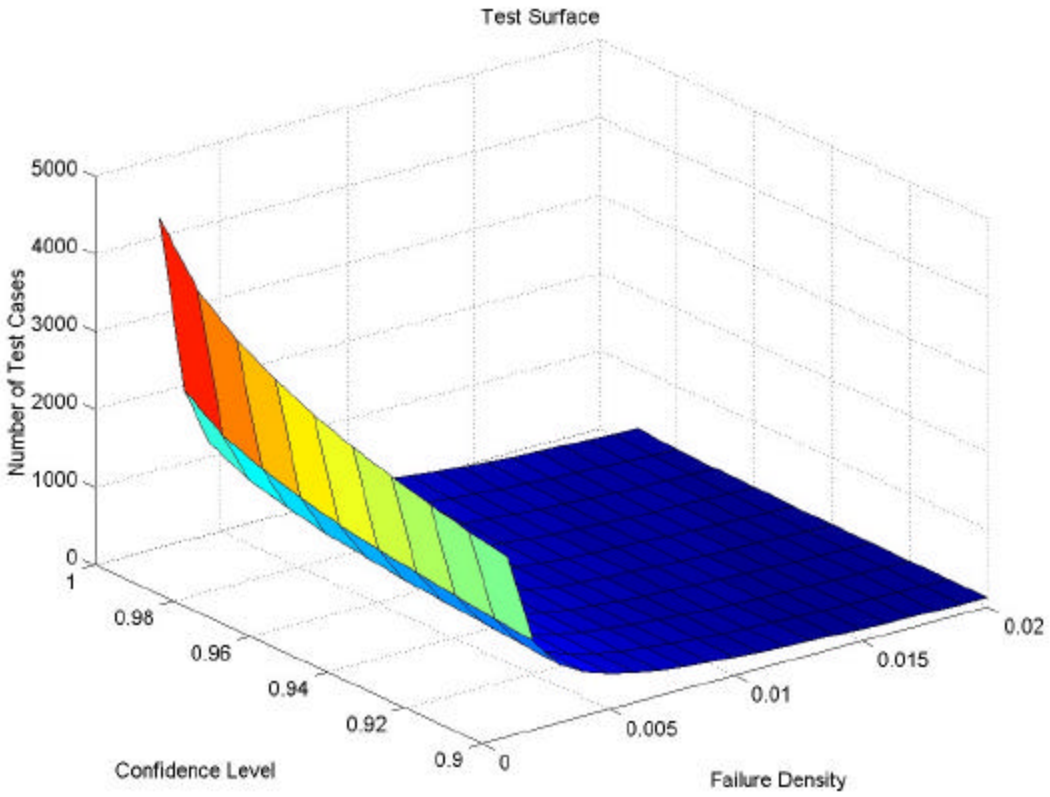


Figure1: Surface of Test Cases versus Failure Density and Confidence Level

	C=0.80	0.82	0.84	0.86	0.88	0.90	0.92	0.94	0.96	0.98	0.99
B=0.01	160	171	182	192	211	229	251	280	320	390	458
B=0.02	80	85	91	97	105	114	125	139	159	194	228
B=0.05	31	33	36	38	41	45	49	55	63	76	90
B=0.08	19	21	22	24	25	28	30	34	39	47	55
B=0.10	15	16	17	19	20	22	24	27	31	37	44
B=0.15	10	11	11	12	13	14	16	17	20	24	28
B=0.20	7	8	8	9	10	10	11	13	14	18	21

Table 1.Number of Test Cases Required for Various *C* and *B* with No Failures

N	*Q*	*B*	*C*		*N*	*Q*	*B*	*C*		*N*	*Q*	*B*	*C*
50	0	0.01	0.39		50	2	0.01	0.01		50	5	0.01	0
50	0	0.05	0.92		50	2	0.05	0.46		50	5	0.05	0.04
100	0	0.01	0.63		100	2	0.01	0.08		100	5	0.01	0
100	0	0.05	0.99		100	2	0.05	0.88		100	5	0.05	0.38
250	0	0.01	0.92		250	2	0.01	0.58		250	5	0.01	0.04
250	0	0.05	0.99		250	2	0.05	0.99		250	5	0.05	0.99
400	0	0.01	0.98		400	2	0.01	0.76		400	5	0.01	0.21
400	0	0.05	0.99		400	2	0.05	0.99		400	5	0.05	0.99
600	0	0.01	0.99		600	2	0.01	0.94		600	5	0.01	0.56
600	0	0.05	0.99		600	2	0.05	0.99		600	5	0.05	0.99

Table 2. Number of Test Cases Needed for various *C*, *B*, and *Q* with Failures

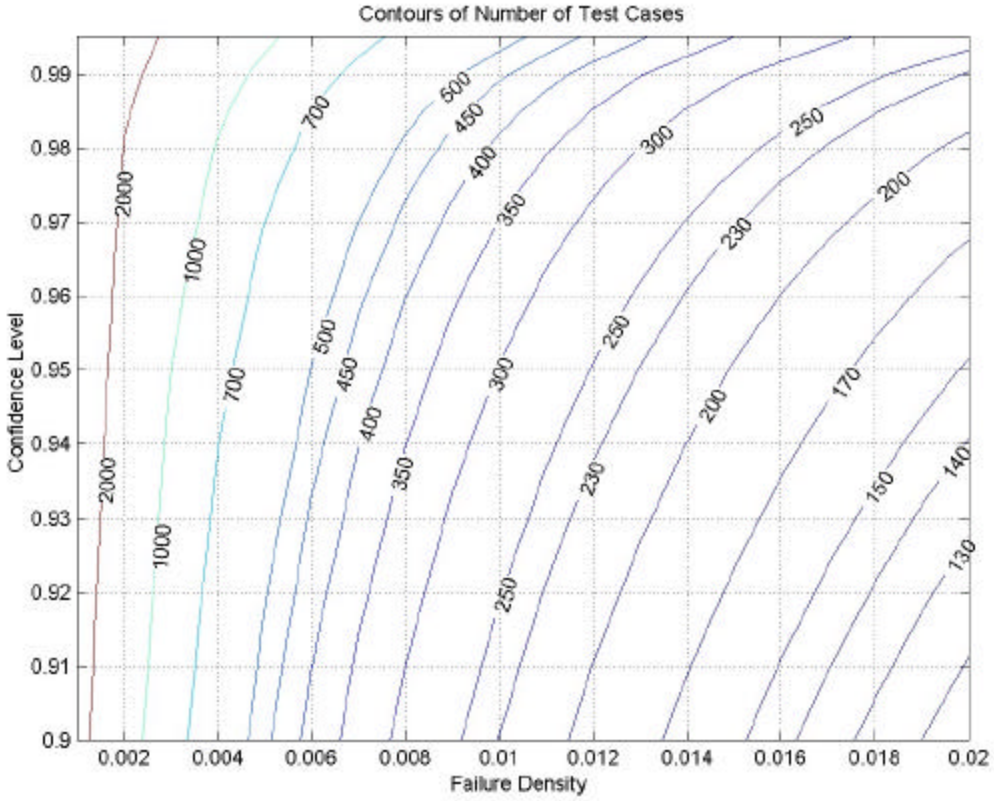


Figure 2: Contours of Test Cases versus Failure Density and Confidence Level

QWE2000 Session 10A

Mr. Bob Bartlett [UK]
(SIM Group Limited)

"A Practical Approach to Testing your eCommerce Web Server"

Key Points

- Since the introduction of the world-wide web, it has never been easier to destroy a company's brand image in such a short period of time. We show you how?
- How do you avoid the problems of too many visitors to your web site?
- It isn't necessary to continuously monitor our site because we check everything before we go home at night.
- Our application is designed for IE so we only need to check it out on IE? Don't we?

Presentation Abstract

How to identify what needs to be tested?

What types of testing need to be performed (Security, performance, functionality, usability and compatibility) when to use continuous testing?

The challenges of automatic notification when problems occur?

Why and when compatibility testing is appropriate?

A practical approach to testing your e-commerce web server

This presentation looks at the super human powers required to ensure that your Web site is capable of the worst that the world has to throw at it through the Internet. It offers some simple guidelines that will enable your organisation to arm itself with an appropriate set of gizmos to keep your e-commerce site on-line taking orders and responsive.

From browser compatibility testing to continuously monitoring your web site. This presentation will show you how, what, where and when to test your e- business application.

About the Speaker

Bob is the Chairman of SIM Group Ltd. SIM specializes in Software Testing and has put in place a number of highly efficient testing systems that automatically test

sophisticated and mission critical software systems. SIM is the UK leader in Providing efficient solutions for software testing. SIM's work has had a profound impact on the way companies approach testing and improvements to testing have been realized with SIM's help. Bob has over 30 years of software experience using automated testing techniques. He is the Executive Director and Chairman of Software testing specialist company today. He is also a member of the CSSA executive council and has designed, developed and sold automated testing tools. Bob, a manager of major software development and implementation projects, is a test adviser to some of the largest testing projects taking place in U.K. Bob has Trained and lectured in automated testing and software testing techniques,has a track record for substantial reductions in time and cost to test, and successfully managed the growth of start up companies throughout his career.

Who we are ….

S I M GROUP GRESHAM

Who are we?

- *Independent testing organisation*
- *Specialists in testing for 10 years*
- *Testing eCommerce systems for the last two years*
- *Largest implementor of automated testing in UK*
- *Full Service testing Solutions*

What do we do?

- *Testing Projects*
- *Hosted Testing*
- *Testing Consultancy*
- *Implementors of automated testing for over 10 years*
- *Methods, procedures and strategies for efficiency & effectiveness*

Agenda

- *Introduction - important considerations for testing eCommerce sites*
- *How to identify what needs to be tested*
- *What types of testing needs to be performed*
- *Compatibility testing - a practical approach*
- *Continuous testing and automatic notification*
- *Summary*

Important considerations for testing eCommerce sites

- *There is an overwhelming temptation to implement - then test:*
  - Time Pressures
  - Ease of testing in the real web environment
  - Simulating realistic loads
- *There is so much that needs to be tested.*
- *We cannot easily ask the users how they intend to use the system*
- *How to protect the risks to the business and the brand.*

Important Requirements for testing ecommerce systems

SIM GROUP GRESHAM

- *24 x 365*
- *Loads impossible to predict*
- *Most sites are an integration of many components from many suppliers*
- *Sites get developed and put in place incrementally and iteratively.*
- *All in the public domain*
- *When something goes wrong - how do you know it has gone wrong?*
- *Some testing WILL happen after the site is live*

How to Identify what needs to be tested

SIM GROUP GRESHAM

- *Application system basics*
- *Interaction with the Environment*
- *Look and Feel*
- *WEB specifics*

Application system Basics

- *Functional*
- *Function and Data security*
- *Load and Stress*

Interaction with the environment

- *Functional*
- *Function and Data security*
- *Load and Stress*
- *Integration*

Look and Feel

- Functional
- Function and Data security
- Load and Stress
- Integration
- Presentation

WEB specifics

- Functional
- Function and Data security
- Load and Stress
- Integration
- Presentation & Usability
- Content
- ISP Operability
- Compatibility
- Site Security

tSpectrum

Looking at a typical project

	Dev.	Link	Sys.	Usab.	Tech	UAT	Oper
Functionality	√	√	√	√		√	
App & Data Security			√	√	√	√	
Load & performance					√		√
Presentation			√	√		√	
Integration		√	√		√		√
Usability				√	√	√	√
Penetration					√		√
Compatibility			√	√	√	√	
Content				√		√	

Who is responsible?

	Owner	Developer	Integrate	Testers	Test Lab
Functionality	√	√		√	
App & Data Security		√		√	
Load & performance					√
Presentation				√	
Integration			√	√	
Usability					√
Penetration					√
Compatibility					√
Content	√				√

Types of Testing

- Functional
- Function and Data security
- Load and Stress
- Integration
- Presentation & Usability
- Content
- ISP Operability
- Compatibility
- Site Security

Functionality testing

- Navigation, menus, links & downloading
- Browser interaction (Back, Forward, Refresh)
- Form filling and data selection
- Submitting transactions and messages (including email)
- Processing
- Calculations
- Negative testing
- Database access
- Handling errors

Security - Function & Data

- *Setting up user access*
- *Data validation*
- *Password assignment, changing and protection*
- *Changing details*
- *Deleting and blocking users*
- *Correct access to site functionality*
- *Protected access to data*
- *Correct audit logs*
- *Negative testing (hacking)*

Load & Stress Testing

- *Establish key customer scenarios*
- *Define volumes, performance profiles*
- *Define Service Level Agreements with ISP*
- *Find realistic data*
- *Automated Testing*
  - Loading the servers
  - Testing across the web
- *Run several times and tune the system*
- *Try to run as early as possible to avoid disappointments*

Integration Testing

- ***Server Integration (WEB, Firewall, presentation, business logic, data base).***
- ***Other site components (Search, help, problem reporting)***
- ***Integration to back office and legacy systems***
- ***Search Engines***
- ***End to end processing***
  - Big Bang
  - Bottom-up combines and tests low-level components into progressively larger modules and subsystems.
  - Top-down combines tests and debugs top-level routines that become the 'test harness' for lower level components.

Presentation & Usability

- *"Look and feel"* Do all of the components of a web site look correct? These components would include; links, tables, forms, controls, images, text and multi-media.
  - ***Inconsistent components.***
  - ***Illogical use of colours***
  - ***Badly aligned components***
  - ***Poor formatting (Fonts, Spacing, word wrap)***
- *"Use of Media"*
- *Target User Group*
- *Intuitiveness and consistency*

Content testing

- ***Component Content - spelling, grammar and relevance***
  - Titles, text, images, banners, links
  - Key text, headlines, section titles & target links
- ***Content Quality and accuracy***
  - Is it correct?
  - Matches the application
  - Changing content
- ***Dynamic content***
  - Source matches display
  - Quality control method

Internet Service Provider (ISP) Operability testing

- ***Matching and tracking to a Service Level Agreement (SLA)***
- ***Robustness and reliability***
- ***Problem handling***
- ***Batch & background processes***
- ***Testing breaks and interruptions***
- ***Handling down times***
- ***Scheduling and staging changes***
- ***Making and testing changes***
- ***Scheduled maintenance***

Site Security Testing

- ***Fully understand set-up and configuration***
- ***Using special software and knowledge - identify potential vulnerabilities - ports, operating systems & other components.***
- ***Test vulnerabilities:***
  - Site Damage
  - Full or partial disable
  - intrusion

Compatibility testing - A practical approach

- ***What you <u>should</u> consider:***
  - Operating System and service packs
  - Browser levels and service packs
  - Connection speeds
  - Monitor types
  - Resolutions
  - Browser options
  - Languages
  - Colour depths
  - Plug-ins

Compatibility testing - A practical approach

SIM GROUP GRESHAM

- ***Know your site***
  - Find out what sort of technology the target users are likely to use
  - Use packages (WebTrends) to find out more about your site
- ***Prioritise tests for greatest usage***
- ***Combine variations to minimise tests***
- ***Stay on top of this for changes***

An example compatibility test covers 98.7% of site accesses

SIM GROUP GRESHAM

Platform	Version	Browser	Resolution	Colour Depth	Connection
Windows NT	SP4	IE401SP1	640x480	256	LAN
Windows NT	SP4	NS4.08	800x600	256	LAN
Windows NT	SP4	IE401SP2	1024x768	65536	56K
Windows NT	SP4	NS4.73	1280x1024	65536	LAN
Windows NT	SP5	IE401SP2	800x600	16777216	LAN
Windows NT	SP5	NS4.5	1024x768	16777216	56K
Windows NT	SP5	IE5.01	800x600	65536	LAN
Windows NT	SP5	IE5.5	1024x768	65536	LAN
Windows 98		IE401	800x600	256	LAN
Windows 98		NS4.08	1280x1024	65536	LAN
Windows 98	SP1	IE401SP2	1024x768	65536	56k
Windows 98	SP1	NS4.5	800x600	16777216	LAN
Windows 98	SE	IE501	1024x768	24-bit	LAN
Windows 98	SE	NS4.73	1280x1024	16777216	56k
Windows 95	OSR2.1	IE4.0	800x600	65536	LAN
Windows 95	OSR2.1	NS4.73	1024x768	16777216	LAN
Windows 2000	Pro	IE5.01	800x600	256	56k
Windows 2000	SP1	NS4.73	1024x768	16777216	LAN
MAC OS9		IE4.5	800x600	65536	LAN
MAC OS9		NS4.72	800x600	65536	LAN

Continuous testing and automatic notification

- ***Just because it worked once - doesn't mean it will carry on working.***
- ***Continuous testing required for:***
  - Functionality
  - Load and performance
  - Security
  - Integration
  - Content (dynamic or changing)
  - Operability
- ***Notify according to severity rules***

Summary

The 12 Commandments

- 1. Speed and performance
- 2. Access and availability
- 3. Up to date and accurate information
- 4. Responsiveness visible and apparent
- 5. Tracking of the business conducted
- 6. Customer service is consistent
- 7. Feedback channels work
- 8. Search and intuitive menus
- 9. Real time processing when beneficial
- 10. Verification of business terms
- 11. Presentation and usability
- 12. Security of site, function and data

Bob Bartlett
bob@simgroup.co.uk
www.simgroup.co.uk

© 2000, SIM Group Ltd..

QWE2000 Session 10I

Mr. Robert L. Probert, Wujun Li, Mr. Paul Sims [Canada]
(School Of Information Technology And Engineering)

"A Risk-Directed E-Commerce Test Strategy"

Key Points

- E-commerce, quality engineering
- Software testing,
- System reliability

Presentation Abstract

E-Commerce frameworks and applications are widely regarded as key engines of an evolving web-based economy. Accordingly, developers and vendors must ensure their quality by utilizing the most effective quality engineering (QE) methods and tools known. At the same time, time-to-market (TTM) constraints and resource limitations require efficient methods, especially in software (functional) testing and system reliability and robustness verification.

In this paper, we present our synthesis of a common test strategy used, often unconsciously, by more effective designers and testers in the software and networking industries, namely Risk-Directed Testing of e-commerce applications and systems. We illustrate its industrial application in two areas, namely

Function Test Reliability and Stress (R&S) Verification.

Finally, we give some empirical observations which support our claims of effectiveness and efficiency, and conclude with a few pragmatic guidelines for refining and improving existing industrial QE processes.

About the Speaker

Robert L. Probert received the Ph.D. in Computer Science from the University of Waterloo in 1973. He is currently a full Professor in the School of Information Technology and Engineering (SITE) and Co-ordinator of the Nortel Networks ASERT (Advanced Software Engineering Research and Training) Laboratory at the University of Ottawa. He is a principal investigator in communications software engineering and protocols for the Communications and Information Technology Ontario (CITO), one of the Ontario Centres of Excellence. He was the first Acting Director of SITE. His research interests and publications are primarily in testing

protocols and networking software. Dr. Probert contributed to International Standards in Conformance Testing, including the conception and prototyping of the TTCN Workbench, a complete environment for test suite engineering. Dr. Probert co-chaired the 10th International IFIP Symposium on Protocol Specification, Testing, and Verification and TestCom 2000, the 13th International IFIP Conference on Testing Communicating Systems. He founded the ACM Symposium on Principles of Distributed Computing, and has frequently collaborated in Software Engineering research with industry and government. In 1989, he received Bell-Northern Research (now Nortel Networks) President's Award of Excellence for work in University-Industry interaction and for Communications Standards work. In 1990, he and his TTCN Workbench team received a TRIO Industrial Feedback award for "creating an innovation of great potential industrial utility". Currently, he has industrial R&D collaborations in progress with IBM, Mitel, Nortel Networks, and Rational Software and is presently a Visiting Research Scientist with the IBM Center for Advanced Studies, specializing in e-commerce testing.

Dr. Probert has given tutorials, workshops and keynote presentations on various topics related to software and system quality engineering, and has designed and delivered training modules for a number of computer and telecommunications corporations. He has taught at the Universities of Waterloo, Saskatchewan and Ottawa, and has consulted with industry and government on a wide variety of topics in the areas of testing, software quality and protocol engineering. He has also been a Visiting Researcher in Software Engineering at GE R&D Center, New York, and various Nortel Labs.

Wujun Li is currently working on network management in Nortel Networks. She received B.Sc.of computer egineering from Beijing University of Aeronautics. She has been working with IBM in the area of E-commerce testing, and finished a Master's thesis in this area in University of Ottawa.

A Risk-Directed E-Commerce Test Strategy

Robert L. Probert
Coordinator, Advanced Software Engineering Research & Training Laboratory
School of Information Technology and Engineering
University of Ottawa
bob@site.uottawa.ca

Behrad Ghazizadeh
WebSphere Commerce Suite System Test
IBM Canada Limited
sims@ca.ibm.com

Wujun Li
Software Design Engineer
Nortel Networks
allyli88@hotmail.com

D. Paul Sims
Project Leader, WebSphere Commerce Suite System Test
IBM Canada Limited
sims@ca.ibm.com

Executive Abstract:

E-commerce frameworks and applications are widely regarded as key engines of an evolving web-based economy. Accordingly, developers and vendors must assure their products' quality by utilizing the most effective quality engineering (QE) methods and tools known. At the same time, time-to-market constraints and resource limitations require cost-efficient methods, especially in product (functional) testing and system reliability and robustness testing.

In this paper, we present our synthesis of a common test strategy used, often unconsciously, by highly effective designers and testers in the software and networking industries, namely Risk-Directed Testing of e-commerce applications and systems. We illustrate its industrial application in two areas:

i) Function Test
ii) Reliability and Stress Verification (emphasis here)

Finally, we give some empirical observations that support our claims of effectiveness and efficiency, and conclude with a few pragmatic guidelines for refining and improving existing industrial QE processes.

Keywords: e-commerce, quality engineering, software testing, system reliability

1. INTRODUCTION AND BASIC DEFINITIONS

Business computer systems, and particularly Internet commerce systems, require a high degree of reliability, dependability, availability, and robustness. Each of these quality attributes must be verified to the degree necessary to avoid exposing the manufacturer to financial, legal, or market (corporate reputation) risks. Thus, e-commerce software, such as IBM® WebSphere™ Commerce Suite, must be well tested with respect to these properties. At the same time, the time-to-market of these products must not be delayed due to test inefficiencies. Therefore, test strategies must involve judicious test case selection to ensure cost-effective risk reduction.

E-commerce products are mainly business-to-consumer or business-to-business. For such products, a primary concern is the avoidance of embarrassing downtime or failures [8]. In other words, reliability, stress, and robustness testing must be designed to cause such failures in the lab, before the product is shipped. If the most likely scenarios (reliability testing) and the highest risk scenarios (stress testing and robustness testing) are verified in the lab, then the customers' perceptions of dependability and availability will be enhanced in the field.

The definitions of key terms such as *scenario* are given below. Informally, a scenario is a sequence of web interactions between the clients and the system, initiated by the clients. A high-risk scenario involves a state or action of client, product, or system component in which a failure may incur a high cost; for example, a web connection is lost in the middle of a payment transaction.

In WebSphere Commerce Suite system testing, the following definitions have been found to be very useful. Many of these definitions are consistent with IEEE definitions [6] and with TPC Benchmark™ W (TPC-W) definitions.

System Under Test (SUT) or simply **System** (see Figure 1):
The *SUT* is composed of all the components that are part of the "application" being tested and that are required to accomplish a specific function or a set of functions. The SUT can reside on multiple server machines and includes all the hardware and software that is in use by the application being tested.

Web Interaction:
A *web interaction* is a complete cycle of communication between the browser and the SUT. This cycle starts when a user selects a navigation option from the previously obtained web page or by typing in a URL. This exchange may include the request and communication of cookies, HTML pages, and client and server side HTTP redirections. The cycle ends when the browser has received the last byte of data from the expected response page.

Server Transaction - or simply **Transaction:**
A *server transaction* is a web interaction that involves some parts of the commerce server on the SUT.

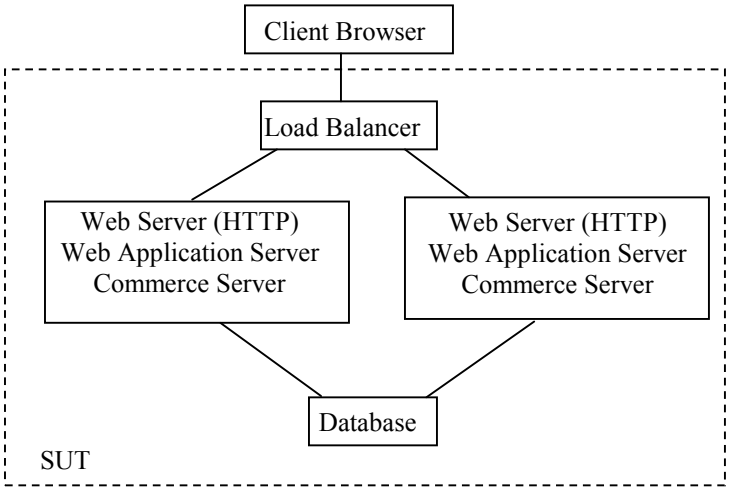


Figure 1. A Typical E-Commerce SUT for SVT

Scenario (User-Oriented View):
A *scenario* is a sequence of web interactions that are logically grouped together because they represent a user task or service. The word *scenario* is often augmented by another word that describes the task the user is trying to accomplish (e.g., shopping scenario means a scenario in which the user is attempting to shop).

Fault:
(1) An accidental condition that causes a functional unit to fail to perform its required function. (2) A manifestation of a design error in software. A fault, if executed, may cause a failure.

Error:
Human design decision or action that results in software that contains a fault. Examples include omission or misinterpretation of user requirements in a software specification and incorrect translation or omission of a requirement in the design specification.

Failure:
An event in which a system or system component does not perform a required function within specified limits. A failure may be produced when a fault is executed.

Scenario Failure:
A *scenario failure* occurs when an unrecoverable failure occurs for any of the web interactions associated with that scenario, or when a recoverable failure occurs three times in a row for the same web interaction in that scenario. Note: After a recoverable failure a user will most likely retry the web interaction. A user will probably retry no more than two times before giving up trying to complete the scenario. This implies allowing three consecutive recoverable failures before the scenario is considered failed.

Function testing (FT) is the process of attempting to demonstrate that functions or features of the SUT do not behave according to specifications, as described in architecture and design documents, in configurations based on realistic customer profiles.

System testing (ST) is the process of attempting to demonstrate that the SUT does not meet its original requirements or set of measurable objectives in complex, high-demand, configurations based on customer profiles. ST covers scalability, load/stress, reliability/availability, performance, security, resource usage, configuration, compatibility/conversion, recovery, serviceability, usability, and installability.

In our experience, the attitude of testing should be to try to elicit failures in the SUT. As Myers [5] states, a successful test case is one that detects the presence of an error or fault by bringing about a failure. The above definitions of function test and system test promote this important attitude. In both kinds of testing, high-yield test cases (those that produce a rich harvest of faults) are preferred. As well, test strategies should be risk directed, as described in the next section.

2. RISK-DIRECTED TEST STRATEGY

Risk is an unwanted event that has negative consequences for someone, usually a *stakeholder*. From a test planning perspective, the degree of risk depends on the point of view of the stakeholder. A stakeholder with respect to an e-commerce system is a party who may experience loss (financial, legal, market) if the system malfunctions in some way.

The major stakeholders in e-commerce testing are the e-commerce software manufacturer, the merchant that purchases and deploys an Internet site using the e-commerce software, and the merchant's customers who shop at the Internet site. In reality, there are many types of stakeholders, including Internet service providers (ISPs), hosting service providers, commerce service providers, and content providers to name a few. We focus on manufacturer, merchant, and customer.

The e-commerce software manufacturer is concerned about all "risks" that can undermine the software quality, including hardware reliability and software capability, usability, performance, reliability, installability, maintainability, security, documentation and service (support) quality.

The merchant can have multiple people performing different roles, and each person may be concerned about different aspects of risk. For example, those involved in site and content creation are most interested in software capability, usability, maintainability, and documentation. Those in technical operations, such as database administrator, system administrator, or store administrator, are likely more concerned about usability, performance, reliability, security, documentation and service quality. Those in business operations, like merchandisers, marketing managers and customer service representatives are concerned about usability, performance, reliability and documentation.

The merchant's customers are most affected by store usability (hence the importance of providing good models upon which customers can base their store design), performance, reliability and security.

The degree of risk that a particular stakeholder may experience during a particular scenario is estimated by the table below:

Frequency of Occurrence	Probability of Failure	Cost of Failure	Risk
L-M	L-M	L-M	L-M
L-M	L-M	H	M-H
L-M	H	L-M	L-M
L-M	H	H	H
H	L-M	L-M	L-M
H	L-M	H	H
H	H	L-M	M-H
H	H	H	H

Table 1: Estimated Risk by Scenario Attributes

In general, a *scenario* is a sequence of web interactions among users and major system components (hardware, software, and network) that has occurred or may occur. Of particular interest in risk-directed testing are the scenarios that have associated high risk according to Table 1.

For example, consider the scenario of a multi-tier e-commerce installation. To assess the risk associated with failure of this type of scenario we consider and rank the risk factors in Table 1, namely frequency of occurrence of this scenario, probability of failure of this scenario, and relative cost of such a failure with respect to a particular class of stakeholder (say, sales and marketing). Here, the frequency of occurrence is *low* (once or twice per customer configuration), the probability of failure is *low* (installability tests are always performed many times before release and the installation process is well-directed by the online installation process), but impact (cost of failure) is *moderate*, since a poor first impression can result in lost future sales. This corresponds to row 1 in Table 1, and thus produces a risk estimate of L-M (low to moderate). Thus this scenario would not normally be included in a risk-directed test strategy, given typical constraints on budget and time-to-market.

As an example of a higher-risk scenario, consider what happens when a web server fails. The frequency of occurrence is low-to-medium, especially for multiple-server configurations, since it is very unlikely that all servers would fail at the same time. The probability of failure is low to medium, since web serving is not an extremely complex task. Finally, the cost of failure is very high, because the site would be rendered unavailable to customers and reputation and revenue would be negatively affected. This scenario corresponds to row 2 of Table 1 with associated moderate to high risk.

Therefore, this scenario should be included in risk-directed testing. A generic risk-directed test strategy is given below.

Generic Risk-Directed Test Strategy

1. Identify stakeholder communities.
2. For each one, identify and rate risks as high, moderate, low.
3. Rank risks × stakeholders = corporate risk.
4. Identify:
   - High corporate risk scenarios.
   - Areas of product that are sensitive to high-risk scenarios.
5. Set scenario and product area coverage targets.
6. Develop tests to cover targets of Step 5 and monitor effectiveness to assure coverage.
7. Monitor field impact and reassess risk assignments, scenarios, and coverage targets.

We were pleased to observe that, in practice, higher risk scenarios were a focus of our functional and system test teams, in many cases without conducting an explicit risk analysis. Perhaps this is due to the inherent tendency of competent testers to attack areas of weakness with high potential negative impact.

In addition to scenarios, testers were quick to identify as high risk any areas of the product that were unstable or had undergone significant code churn during development, or that implement complex business rules, or that have a complex (large) implementation as product areas requiring test coverage [1].

In the next two sections we briefly give some function test principles and then focus on system testing (stress and reliability testing).

3. HIGH-YIELD, RISK-DIRECTED FUNCTION TESTING

According to Myers [5], the best testers strive to uncover errors. The *high-yield strategy* defines key scenarios based on a customer orientation, which will cost-effectively detect the most errors in design and development [8]. The cost effectiveness comes by selecting types of scenarios, called high-yield scenarios, which are likely to cause design errors to be manifested in design simulations or inspections and walkthroughs. Very specifically, by yield, we mean the number of dangerous errors or high-risk errors that are detected by walking through, simulating, or executing this scenario. For example, an error of very low associated cost that occurs frequently enough to annoy a customer may be considered a moderate to high-risk error. Similarly, an error that occurs very infrequently but whose associated cost is enormous may also be classified as moderate to high risk. Scenarios are classified as high yield, moderate yield or low yield scenarios.

Since scenarios have been selected according to the likelihood of yield of high-risk errors, it is important to monitor and measure the degree of coverage of risk associated with scenarios that are selected for use cases and customer requirements from the very beginning of a project. The metrics are computed from measures that are made during

the requirements analysis and high-level design development process. Requirements are organized by functional area. Each functional area will be covered by scenarios. The first two measurements that are needed are the number of requirement areas by function and the total number of scenarios.

Next, functional testers or designers identify scenarios that are usually extremely well-understood by all stakeholders in the system design and development of the process, particularly customer representatives, owners, developers, designers, and testers. These are designated low-yield (L). The second category of scenarios is moderate to high-yield scenarios. For many of these, the proper system reactions are unknown and certainly are not uniformly understood. Differences in interpretations of requirements, given these particular scenarios, will lead to invalid design assumptions and design omissions. The third primary measurement is the number of low-yield scenarios. After these primary metrics have been computed, then the additional metrics listed below can be calculated and used to guide the requirements and design quality assessment process. Table 2 gives the formulas for calculating these coverage assessment and risk management metrics.

Name of Metric	Formula
Number of requirements for functional areas	Primary measurement (a)
Total number of scenarios	Primary measurement (b)
Number of low-yield scenarios	Primary measurement (c)
Number of high-yield scenarios	d = b - c
Average basic coverage	e = b / a
Average risk coverage	f = d / a
Degree of risk orientation	g = d / b
Scenario risk ratio	h = d /c

Table 2. Key Metrics Formulae

Example Ordering System:

In this example, we apply our high-yield requirements capture strategy to an electronic commerce application involving an online ordering system. First, consider a use case of the requirements specifications, namely, online ordering. Then, we classify the scenarios according to our classification approach. Finally, we apply our key metrics formulae to assess the risk-directed functional test coverage.

Online ordering use case:

This use case mainly deals with the customer going online to order one or more items from a company's business catalogue. The main purpose of this use case is to successfully place an order through the system, which eventually results in the customer receiving the ordered items and being billed for the total price. There is only one low-yield scenario, which involves browsing the catalogue, paying online and receiving all

necessary confirmations. We have also identified five moderate-yield scenarios in which some errors occur but eventually are corrected by the customer, therefore resulting in a successful completion of the use case. Fourteen high-yield scenarios were identified in which some errors occur, but the customer either fails to fix them or decides to cancel, therefore resulting in the unsuccessful termination of the use case. Finally, we identified eleven concurrent high-yield scenarios of which four result in the successful termination of the use case and the rest do not lead to a successful completion of all the concurrent order operations.

Table 3 below summarizes the results of applying our metric formula on the use case described previously.

Name of Metric	Formula	Use case: Online ordering
Number of requirements for functional areas (use cases)	a	1
Total number of scenarios	b	36
Number of low-yield scenarios	c	11
Number of high-yield scenarios	d = b - c	25
Average basic coverage	e = b / a	11/1 = 11
Average risk coverage	f = d / a	25/1 = 25
Degree of risk orientation	g = d / b	25/36 = 0.69
Scenario risk ratio	h = d / c	25/11 = 2.27

Table 3. Metrics Obtained for Online Ordering

We can notice that our approach is biased toward the problem areas of the requirement specifications since the ratio of high-yield/low-yield is well over 1. The effect of this bias will not only be seen at the specification and design phases of the development process; if these scenarios are used as the basis for test suite generation, more serious errors will be caught [7]. Moreover, if these high-yield scenarios are considered during requirements capture, the likelihood of having related errors decreases considerably. We feel having more high-yield scenarios earlier in the process will improve the software quality and coverage in terms of requirements capture and testing. If we have a low ratio, i.e., more low-yield scenarios, we risk wasting test time and missing potential high-cost errors.

A benefit of this approach is that the high-yield scenarios evolve into very effective test cases. The time to market is lessened because the functional (customer-oriented) tests are developed in parallel with design and development instead of waiting until all code is produced. In addition, significant problems will be manifested at the design stage where they can be corrected with a minimum of redesign and no redevelopment.

The only shortcoming to the high-yield approach is that we have not yet built a "smart spreadsheet" support tool to ease the bookkeeping tasks for the measures and metrics. This is a straightforward task.

4. RISK-DIRECTED SYSTEM TESTING (RELIABILITY & STRESS)

Reliability is one of the e-commerce quality factors most often cited by clients, and therefore is a key goal for e-commerce test groups. The following are benefits of reliability testing:

- Customer Scenarios: Reliability testing actually executes a real customer's (usually a large customer's) load, and produces a quantitative measurement of the reliability of the system. This metric can enable useful comparisons with other IBM products and with competitors' products. This information can then be used to compare the reliability of the application with previous versions and other applications or components.
- Customer Loyalty: Functionality and price often convince a customer to buy a certain product but it is usually the reliability of the product that keeps customers. This has been demonstrated in many other industries, a prominent one being the automobile industry.
- Integrated System: Since we ship WebSphere Commerce Suite with many component products that are developed and tested elsewhere, it is very important for us to test the reliability of the entire system, as the interaction in this complex environment could be the source of many reliability problems (e.g., many system interaction problems were discovered by reliability testing).

We now give some background on reliability metrics and testing.

Reliability is the ability of the SUT to perform its required functions under stated conditions for a specified period of time. These stated conditions are usually customer-specific (i.e., reliability testing validates the ability of the SUT to perform its required functions under a typical customer load). The main variable in reliability testing is the period of time (the duration) for which the test is executed. As Musa and others have pointed out, the validity of test results for prediction relies mainly on the accuracy of the customer usage profile [11].

Previous approaches to reliability testing focused on measuring mean-time-to-failure (MTTF), mean-time-between-failures (MTBF), and mean-time-to-repair (MTTR). These metrics are defined as: [3,7]

i) MTTF of a system over a test period is the average of the interfailure times during that period,
ii) MTTR of a system over a test period is the average time to diagnose and repair faulty software components,
iii) MTBF is MTTF + MTTR.

Reliability can be measured as MTBF/(1+MTBF) or as failure intensity (FI), which is the rate of arrival of failure (failures per CPU hour) over a test session. Availability can be measured as the likelihood that the system is operational, or MTBF/(MTBF+MTTR). Reliability is usually based on CPU time; availability is estimated with respect to elapsed time.

Reliability testing generally means running the system for 24 to 72 hours under a realistic, moderate to heavy load. It is important that all reliability testing use a representative workload (operational profile).

The FI metric is useful for plotting trends over test time towards meeting a "failure intensity objective" before release. For example, an FI objective of .04 means that failures may arrive at the rate of no more than 1 failure for every 25 CPU hours. In general, CPU hours should be used because that measure is more reliable than clock hours. In practice, many organizations use clock hours. In such a case, if FI * .04 with respect to clock hours, we would be expecting no more than one failure occurring per day.

In our risk-directed strategy, we set up five "hurdles" for pure reliability testing, one each for 12, 24, 36, 48, and 72 hours. This is most efficient if serious functional defects have already been caught by the Function Test team, and repaired correctly. In practice, of course, some functional failures will be observed [2].

Stress Testing

Stress testing is managed according to a *monotonic hurdles* approach wherein the system is required to meet increasing levels of throughput (scenarios per hour) and concurrency (number of simultaneous virtual users). This is illustrated in Figure 2.

It is important to note that each hurdle does not have to necessarily map one-to-one to a test case. For example, in Figure 2, we could have a test case defined that will run 64 concurrent users and have a throughput of 400 scenarios/hour. This will automatically cover two hurdles, one on each of the throughput and concurrency hurdles charts. Furthermore, if at the beginning of testing the test case corresponding to the final step was executed successfully, then there would be no reason to execute the remaining test cases since having 6400 scenarios/hour and 256 concurrent users is inclusive of all other hurdles. However, success of the top hurdle on initial tries usually points to the fact that the hurdles were not chosen high enough.

With this approach, if there are defects that are blocking some of the top steps, then the tester will make progress on the bottom steps and discover the upper limit of the system. This adds to the ability of the management team to make better decisions about whether the product is ready to be shipped.

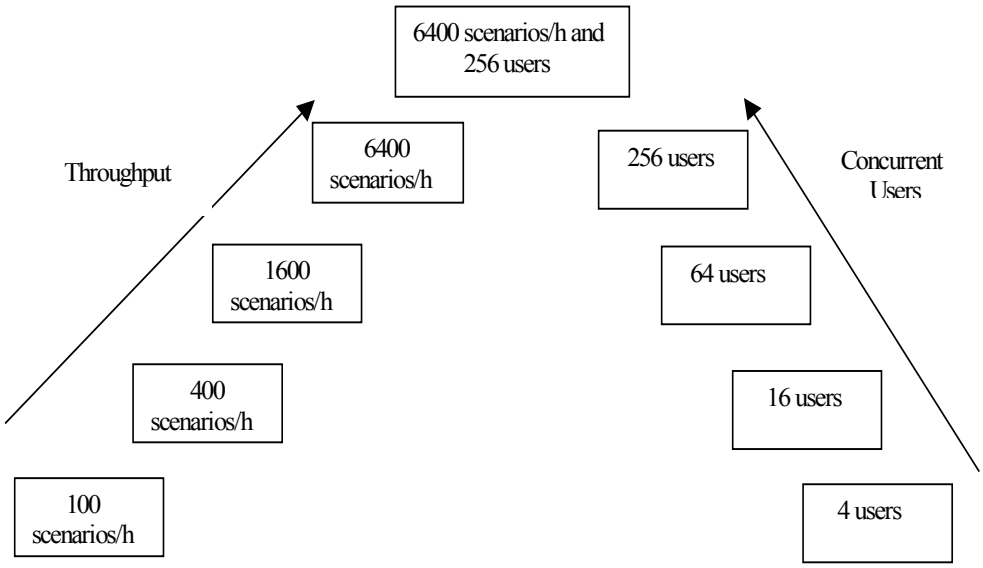


Figure 2. Stress and Reliability Hurdles

Selecting Appropriate Hurdles

The choice of which values of throughput and concurrency should represent each hurdle and of which minor hurdles should be tested are left to the discretion of the component test plan writers. However, the hurdles' values must achieve at least 200% of the load requirements as set out by the requirements document for the application being tested. Also, starting with values considerably under the requirements may be an advantage in driving out defects. The newer the component (or the less it has been stress tested in the past) the lower the initial hurdle should be.

For example, in one product, store creation was the type of scenario tested. The highest hurdle created about 10,000 stores (scenario throughput of 10,000 stores per day). This is more stores than what most customers want to create in one year. Thus, the highest hurdle from a throughput perspective was 365 times that of the requirement. The lowest hurdle for the concurrent virtual users was set at two early in the test cycle.

5. OBSERVATIONS AND PRAGMATIC GUIDELINES FOR IMPROVING THE DEVELOPMENT AND TEST PROCESS

Test Involvement

Test should be involved in the development cycle from the beginning. Test should verify that product requirements are accurately reflected in programming objectives, architecture, and design specifications, and ultimately validate running code. The sooner an error or design issue is detected, the less it costs to remove from the product.

Stress and Reliability Tests

Stress tests run for short periods of time, usually not exceeding three hours. Stress tests determine the behavior of the SUT under high load, which can be simulated by increasing throughput and the number of concurrent users. Load tests should attempt to achieve at least twice the specified throughput and number of concurrent users. Lower values are initially chosen and often identify defects early in the test cycle. Load is increased when a test scenario runs successfully until twice the specification is attained or the SUT fails. Error statistics are measured in two ways:

i) Web interaction failure ratio, (number of single web page hits failing)/(total number of requests), and

ii) Scenario interaction failure ratio (e.g., counting the rate of failure of entire shopping scenarios).

A stress scenario is considered to have run successfully when results fall within a range of expected values for web interaction response time (e.g., not exceeding two seconds), web interaction failure ratio (e.g., not exceeding 1%), and scenario failure ratio (e.g., not exceeding 3%). Stress tests precede reliability tests.

Reliability tests run for longer periods under loads representing typical customer use. The load can vary over the duration of the reliability test, which usually lasts not less than 12 hours and can run many days. A reliability scenario is considered to have run successfully when results fall within a range of expected values for web interaction failure frequency (e.g., not exceeding five per hour) and scenario failure frequency (e.g., not exceeding one per hour) over the duration of the reliability test.

Stress and reliability test results are summarized and compared to previous releases and to other products and components. The goal is to always achieve continuous reliability improvement from release to release.

6. CONCLUSIONS

Without going into proprietary documentation to give the details of risk identification, it was generally found that it was beneficial for the verification test effort to:

i) Focus on finding defects.

In e-commerce, this assumes that defects are most prevalent in areas that other test teams would not have explored. This would often be in the integration of all the various (add-on) components, and also the product runtime environment, where stress and scalability tests would "shake out" high-load/concurrency/reliability defects which would not appear in functional testing.

ii) Use a risk-directed test strategy.

Use a simple risk analysis to identify high-risk scenarios according to Table 1, and ensure that these scenarios are covered during function test, and during system stress and reliability tests. We have found it beneficial to test highest-risk scenarios first. This avoids wasting time on scenarios that are unlikely to occur or have low cost of failure.

iii) Use test automation tools.

Finally, test automation is extremely helpful in both function test and in reliability and stress tests. A sample automated test architecture is shown in Figure 3.

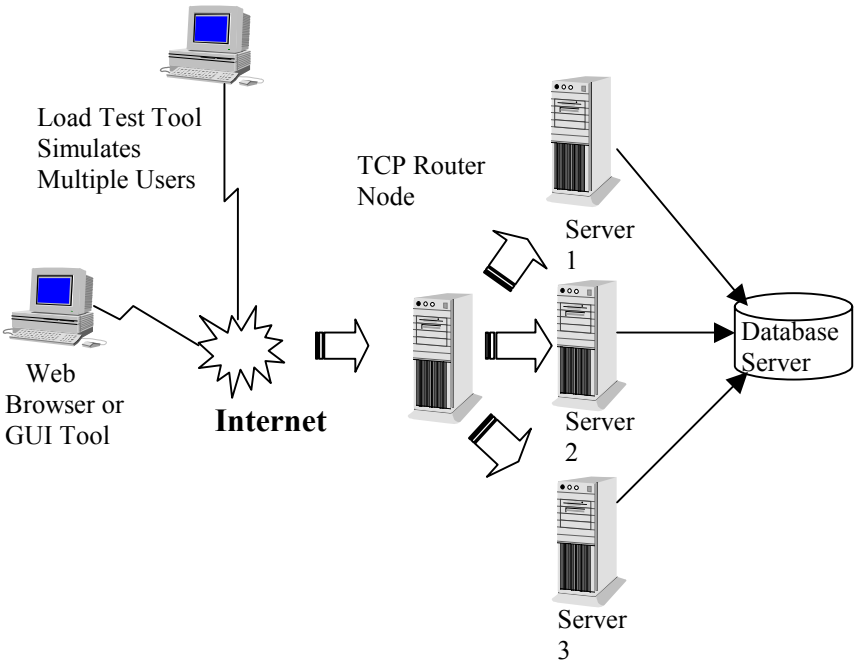


Figure 3. System Test Configuration

At the outset of the current release of IBM WebSphere Commerce Suite, the system verification test team was challenged by management to reduce its cycle time by 30 percent. A shorter test cycle was required to stage the product's release across multiple operating systems and to accelerate time-to-market. Knowing that it would not be possible to sustain the level of test effort that had been applied to previous releases, the team defined its test strategy and coverage accordingly. While it is too early to judge the effectiveness of our approach, it appears that the risk-directed strategy will result in significantly fewer high-risk defects over the product's life-cycle. We strongly recommend this approach to other e-commerce test groups.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of this work by Communications and Information Technology Ontario and by IBM Canada Limited.

The views expressed in this paper are those of the authors and not those of IBM Canada Limited.

The following are trademarks or registered trademarks of International Business Machines Corporation: IBM, WebSphere.

References

[1] R. Binder, Scenario-Based Testing for Client-Server Systems, Software Testing Forum, 1993, 1,2, 12-17.

[2] IBM Internal Documentation

[3] S.H. Kan. Metrics and Models in Software Quality Engineering, Addison-Wesley, 1995.

[4] D.J. Mosley. Client-Server Software Testing on the Desktop and the Web, Prentice-Hall, 2000.

[5] G.J. Myers. The Art of Software Testing, John Wiley & Sons, 1979.

[6] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std. 610, 12 (1990).

[7] S.L. Pfleeger. Software Engineering: Theory and Practice, Prentice Hall, 1998.

[8] K. Saleh, R.L. Probert, and W. Li. High-Yield requirements capture for electronic commerce software. *International Symposium on Electronic Commerce and the 2nd International Workshop on Technological Challenges of Electronic Commerce*, Beijing, China, May 1999.

[9] K. Saleh and R.L. Probert. Issues in Testing E-commerce Systems. Electronic Commerce Technology Trends, challenges and Opportunities, ed. W. Kou, Y. Yesha, IBM Press 2000. 273-282.

[10] G.W. Treese and L.C. Stewart, Designing Systems for Internet Commerce, Addison-Wesley, 1998.

[11] J.D. Musa, Software Reliability Engineering, McGraw-Hill, New York, 1998.

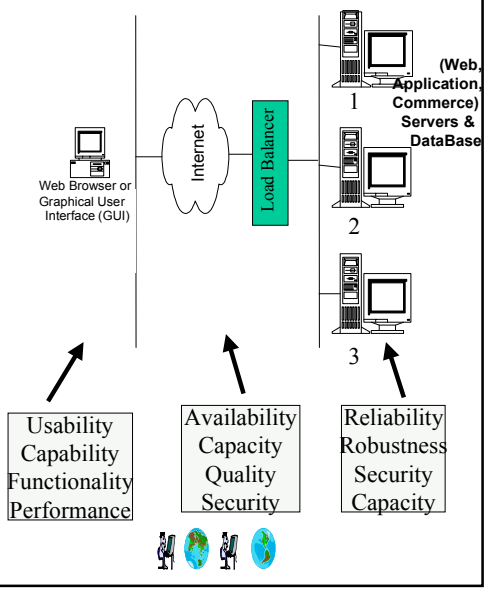
A Risk-Directed E-Commerce Test Strategy

Robert L. Probert, Coordinator
Advanced Software Engineering Research & Training Laboratory
School of Information Technology and Engineering
University of Ottawa
bob@site.uottawa.ca
Paul Sims, Project Leader, Behrad Ghazizadeh
WebSphere Commerce Products System Test
IBM Canada Ltd.
sims@ca.ibm.com
Wujun Li, Software Design Engineer
Nortel Networks
allyli88@hotmail.com

Outline of Presentation

- Intro to E-Commerce Testing
- Basic principles of Risk-Directed Testing
- High-yield, risk-directed Function Testing
- Risk-directed System Stress and Reliability Testing
- Observations & Pragmatic Guidelines
- Conclusions & Recommendations

Intro to Testing E-commerce



Introduction to E-Commerce Testing (cont)

- **Key Definitions:**
  - SUT
  - Web Interaction
  - Server Transaction
  - Scenario (user-oriented view)
  - Error
  - Fault
  - Failure
  - Function Test (WITH ATTITUDE)
  - System Test (WITH ATTITUDE)

Risk-Directed Testing

- Risk
  - "an unwanted event with negative consequences for a *stakeholder*"
- stakeholders
  - manufacturer
  - merchant (multiple roles)
  - customer

Estimating Risk by Scenario Attributes

FO PF CF R

Frequency of Occurrence	Probability of Failure	Cost of Failure	Risk
L-M	L-M	L-M	L-M
L-M	L-M	H	M-H
L-M	H	L-M	L-M
L-M	H	H	H
H	L-M	L-M	L-M
H	L-M	H	H
H	H	L-M	M-H
H	H	H	H

Examples:

Row 1: Multi-Tier Installation

Row 2: Web Server Fails

Generic Risk-Directed Strategy

1. Identify stakeholder communities.
2. For each one, identify and rate risks as high, (moderate), low.
3. Rank risks x stakeholders = corporate risk.
4. Identify:
   - High corporate risk scenarios.
   - Areas of product that are sensitive to high-risk scenarios.
5. Set scenario and product area coverage targets.
6. Develop tests to cover targets of Step 5 and monitor to assure coverage.
7. Monitor for field impact and re-assess risk assignments, scenarios, and coverage targets.

High-Yield FunctionTest Strategy

- Origen:Art of Software Testing [G. Myers]

  (1) one cannot test a program to guarantee that it is error-free and

  (2) a fundamental consideration in program testing is one of ***economics***

  Since exhaustive testing is out of the question, we must maximize YIELD on the testing investment (i.e., maximize the number of errors found by a minimum cost set of effective test cases).

What is High-Yield? … A High-Yield Attitude?

- **Yield** is the number of serious defects detected in the product
- **High-yield** implies a rich harvest of bugs!
- A **high-yield attitude** is
  "I know there are lots of serious bugs here, and I am going to find them!"

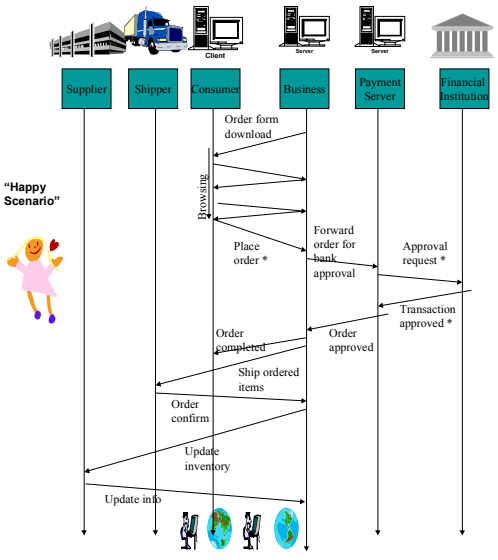
Can Metrics Help?

Yield -Based Risk-Directed Coverage Metrics

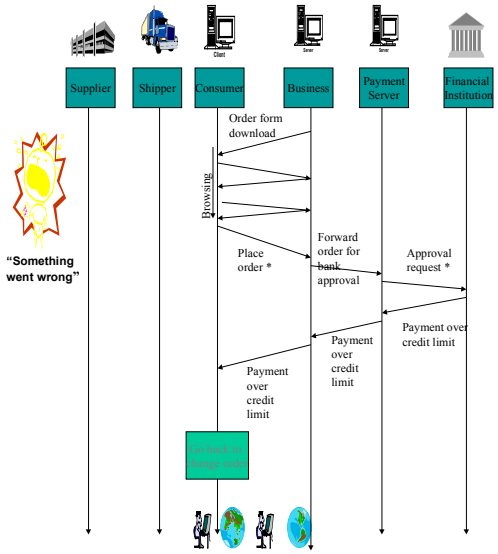
Name of Metric	Formula
number of requirements for functional areas	primary measurement (a)
total number of scenarios	primary measurement (b)
number of low-yield scenarios	primary measurement (c)
number of high-yield scenarios	d = b - c
average basic coverage	e = b / a
average risk coverage	f = d / a
degree of risk orientation	g = d / b
scenario risk ratio	h = d /c

Example of Scenarios

Low-Yield scenario



Example of Scenarios
High-Yield scenario



Example Metrics for on-line ordering

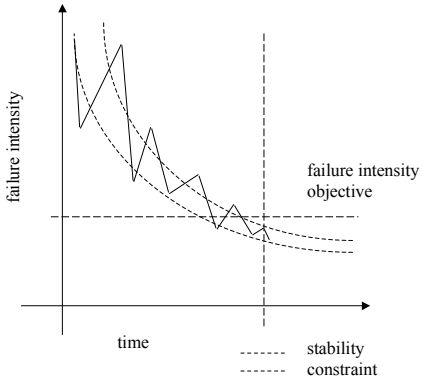
Name of Metric	Formula	Use case: Online ordering
Number of requirement for functional areas (use cases)	a	1
Total number of scenarios	b	36
Number of low-yield scenarios	c	11
Number of high-yield scenarios	d = b - c	25
Average basic coverage	e = b / a	11/1 = 11
Average risk coverage	f = d / a	25/1 = 25
Degree of risk orientation	g = d / b	25/36 = 0.69
Scenario risk ratio	h = d / c	25/11 = 2.27

NOTE: 2 H-Y FOR EVERY L-Y SCENARIO

Risk-Directed Stress & Reliability Testing

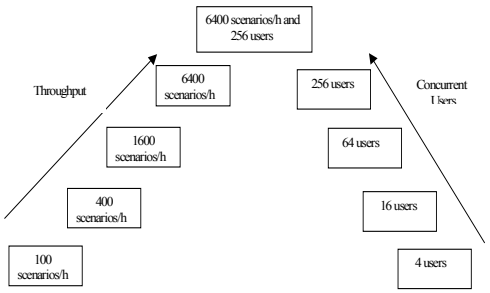
- Reliability Metrics
- Reliability Testing
- Stress Testing Strategy
- Example of Hurdles

Increase Yield by Realistic Testing (theory)



key metric: failure intensity

Stress and Reliability Hurdles

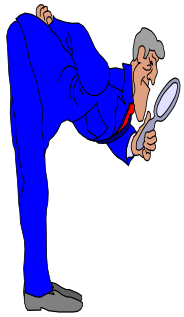


Observations and Pragmatic Guidelines

- Error Statistics and Targets
  - e.g., for stress testing (hours)
    - web interaction response < 2 sec.
    - web interaction failure ratio < 1%
    - scenario failure ratio < 3%
  - e.g., for reliability testing (days)
    - web interaction failures < 5 per hr.
    - scenario failure frequency < 1 per hr.
- Run Stress Tests BEFORE Reliability Tests
- Stress to double throughput & number of concurrent users

Conclusions and Recommendations

- **1. Attitude**:
  Focus on Finding Errors



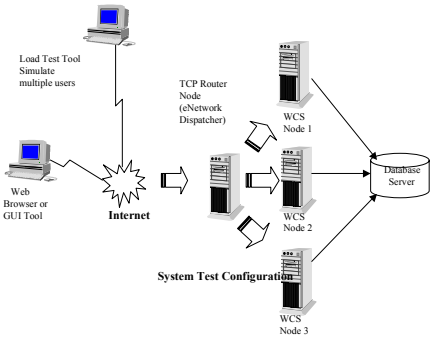
Recommendations (cont)

2. Use Risk-Directed Strategy

Test Highest Risk Scenarios First

Recommendations (cont)

3. Automate where it is most effective and useful



Good Tools are Available

SUMMARY

- **Quality & Productivity** improvements are incrementally achieved by a **Risk-Directed, High-Yield Strategy**
  - finds errors early
  - catches most severe errors
  - maximizes ROTI (return on test investment)

Acknowledgements

- Communications & Information Technology Ontario
- IBM
- Natural Sciences & Engineering Research Council of Canada

Thank You!
Questions??
Comments??

QWE2000 Session 10M

Mr. William E. Lewis [USA]
(Technology Builders)

"A Continuous Quality Improvement Testing Methodology (10M)"

Key Points

- Review the waterfall approach to software development
- Review the V-model approach to software development
- Learn how verification and validation testing is performed using the V-Model
- Understanding the problems encountered in software waterfall development
- Learn the characteristics of rapid application development
- How to apply Deming's quality improvement principles for generically
- How to apply Deming's quality principles to the rapid application development (RAD) development approach
- Contrast the psychology of testing between the waterfall and RAD development
- Learn the phases, tasks and steps of the continuous quality improvement testing methodology

Presentation Abstract

Rapid application development (RAD) methodologies are a reaction to the traditional waterfall systems development in which a product evolves in sequential phases. A common problem with the life cycle development model is that the elapsed time to deliver the product can be excessive with user involvement only at the very beginning and very end. As a result, the system that they are given is often not what they originally requested.

By contrast, RAD development expedites product delivery. A small but functioning initial system is built and quickly delivered, and then enhanced in a series of iterations. One advantage is that the users receive at least some functionality quickly. Another advantage is that the product can be shaped by iterative feedback, i.e. users do not have to define every feature correctly and in full detail at the beginnning of the development cycle, but can react to each iteration. One common mistake is that during the process the requirments are not documented during each iteration and the does not exist as a "final" requirements document, just the system.

RAD or spiral testing is dynamic and may never be completed in the traditional sense of a delivered system's completeness. The term "spiral"

refers to the fact that the traditional sequence of analysis-design-code-test phases are performed on a micro scale within each spiral or cycle in a short period of time, and then the phases are repeated within each subsequent cycle.

Learn how to apply Dr. Edwards Deming's continuous quality improvement techniques which where originally used in the manufacturing setting. They are applied as a continuous quality improvement testing methdology which is superimposed over Deming's plan-do-check-act quality wheel.

About the Speaker

Bill Lewis has 35 years experience in the computing industry. Currently as a senior technology engineer he trains and consults in the requirements-based testing area which focuses on leading-edge testing methods and tools. He teaches Writing Testable Requirements, Requirements-Based Testing, Ambiguity Reviews, Reviewing Requirements Using RM and numerous seminars. He is also an active client practitioner of TBI's Caliber-RBT, a requirements-based functional test case design tool.

Before joining TBI, he was an assistant director for Ernst & Young, LLP for 6 years as the quality/ testing manager for several E&Y application development projects. He was also the senior project manager for the ISO9000 project resulting in a successful international certification. Bill authored several technical and methods development handbooks for E&Y. Prior to that he was a quality analyst in Jeddah, Saudi Arabia for the Saudi Arabian Oil Company (ARAMCO).

The majority of Bill's career was at IBM for 28 years. His jobs included system programmer, analyst, performance analyst and technical instructor. With IBM, Bill has consulted and trained all over the world, including Amsterdam, South Hampton, Toronto, Rome, Seoul Korea, Hong Kong, Thailand, Singapore, Sydney, Australia, and the U.S.

His first job out of college was with the Apollo Support Department for General Electric at the Kennedy Space Center as a real-time programmer for the Apollo project. After completing his service committment, he worked for Radiation, Inc. as a real-time programmer on the Nimbus-D satellite program.

Bill is a prolific communicator having lectured at various quality organizations including the Quality Assurance Institute (QAI) Fourth International Quality Conference, the American Society for Quality, and Association of Information Technology Practitioners. His has also taught computer courses as a part-time adjunct professor for five years and authored five books on computer problem solving. In 2000 he recently authored a book entitled "Sofware Testing and Continuous Quality Improvement" which is the basis of this seminar.

Bill holds a BA degree in Mathematics from the University of Miami, Florida

and an MS in Operations Research from the University of Central Florida. He is also a Certified Quality Analyst (CQA) and Certified Software Test Engineer (CSTE) through the Quality Assurance Institute.

A Continuous Quality Improvement Testing Methodology

Bill Lewis
Senior Technology Engineer, CQA, CSTE

Technology Builders, Inc.
400 Interstate Parkway
Suite 1090
Atlanta, Georgia 30339
tel: 770-937-7900
fax: 770-937-7898
http://www.tbi.com

Software Testing and Continuous Quality Improvement

Topics

- Deming's Quality Improvement Framework
  - Role of statistics and metrics
  - The quality wheel (PDCA)
- Waterfall Development Versus RAD Development
  - Phased approach
  - Problems and limitations
  - Psychology of testing
- What is RAD/ Spiral testing?
  - A continuous quality improvement testing methodology

Software Testing and Continuous Quality Improvement

Role of Statistics and Metrics - Deming

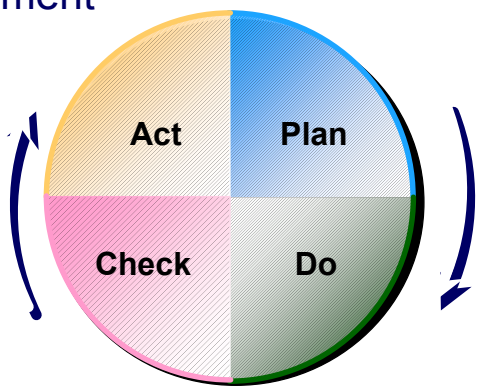
- Deming has influenced every facet of work in every industry, influencing government, schools, hospitals.
- "In God we trust. All others must use data"
- Statistics is a tool to help us Understand, Gain Control, and Monitor a process.
  - Flow Chart (understand a process by flow charting the process)
  - Run Trend Chart (plots data points in chronological order)
  - Control Chart (run chart with statistical lower/upper bounds around average)
  - Cause-and-Effect (identifies possible causes of a problem)
  - Histogram (data organized according to relative frequency)
  - Pareto Chart (incidents ranked by frequency in ascending order - 80/20 rule)
  - Scatter Diagram (charting the relationship between two variables)

Deming's Quality Principles

- Quality Points
  - Statistics alone are not enough
  - Need bedrock of management philosophy (14 points)
- Seven Deadly Diseases
  - Lack of constancy of purpose
  - Emphasis on short-term profits
  - Evaluation by performance and merit ratings
  - Mobility of management
  - Running a company on visible figures alone
  - Excessive costs
- The continuous quality improvement wheel
  - Continuously improvement through Plan-Do-Check-Act

PDCA Quality Wheel -

a quality framework for continuous improvement



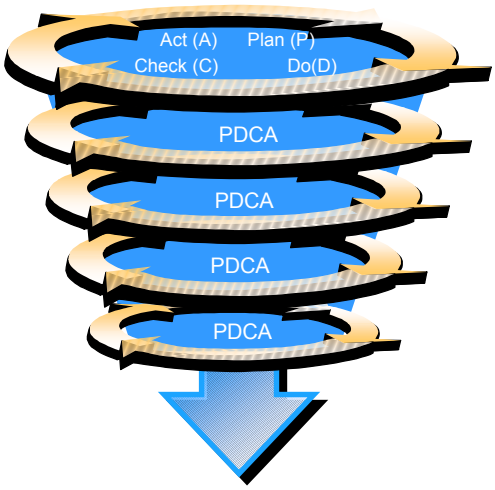
- **Plan**
  - Define the quality objectives and determine the required conditions and methods required to achieve the objectives
  - Clearly define and quantify the objectives
- **Do**
  - Create the conditions and perform the necessary actions to execute the plan
- **Check**
  - Determine if work is progressing according to the plan and expected results are obtained
  - Compare the results with objectives
- **Act**
  - Devise measures for appropriate actions if work is not performed according to the plan or results are not achieved

Software Testing and Continuous Quality Improvement

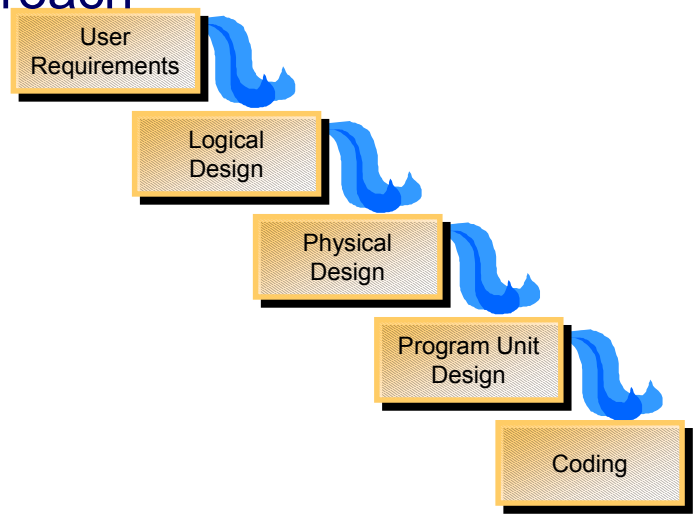
Iteration of the P-D-C-A Wheel

- PDCA Approach Ensures
  - Quality of products
  - Delivery date
  - Expected costs
- Repeated PDCA Usage Continuously Improves
  - **Quality of work**
  - **Work processes or methods**
  - **Repeatability**
  - **Deliverables or results**
  - **People**
  - **Technology**

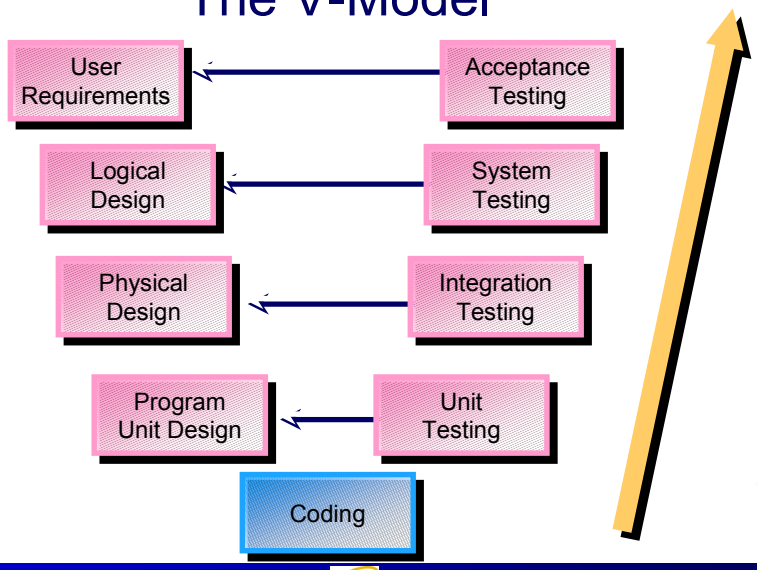
PDCA Ascending Spiral



Waterfall Development - linear approach



The V-Model



Software Testing and Continuous Quality Improvement

Defining the V-Model

- Verify ensures project deliverables are complete and comply with standards (required documents, tasks)
- Validate ensures business and technical requirements (inputs, outputs) have been correctly translated into deliverables (all requirements from previous stage accounted for and correctly represented)
- Test Planning defining test conditions and requirements concurrently with requirements from specification phases
- Test that completed programs and systems deliver the requirements

Waterfall Development Limitations

- Long Development Cycle and Shortened Business Cycles
  - Results in a gap between requirements and delivery
- End users Involved in the Very Beginning and Very End
  - Expected to define requirements in detail
- "Rippling Effect"
  - Return to previous phases which is very costly
- Software testing often treated as a separate phase
  - Often late involvement (testing the final system)
- Practitioners started looking at alternative approaches
  - RAD or Spiral Development, prototyping

What is RAD?

- Process of working from a base and building a system incrementally in "spirals" (spiral development)
  - Small but functioning initial system is built and quickly delivered, and then enhanced in a series of iterations
  - Rapid prototyping
  - "Hacker RAD" is not RAD
- Advantages
  - Users receive functionality quickly
  - Product is shaped dynamically by iterative feedback
- Disadvantage
  - Tendency NOT to document requirements as you go!
  - The product definition and specifications continue to evolve indefinitely, e.g. there is no such thing as a frozen specification--> how to test it?

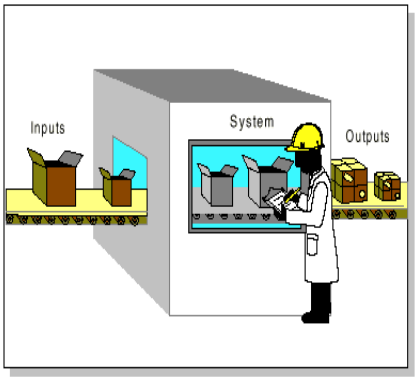
Psychology of Testing

- Waterfall Development
  - Requirements defined (assumption)
  - More efficient for outsiders test the code
  - Developers should NOT test their own work
  - Does not work in a spiral development environment
- RAD Development (spiral)
  - Requirements will not be totally defined
  - Cooperation between testers and development is imperative
  - Testers are powerful allies
  - Test and development management commitment

Testing as a "Zero Sum" Game

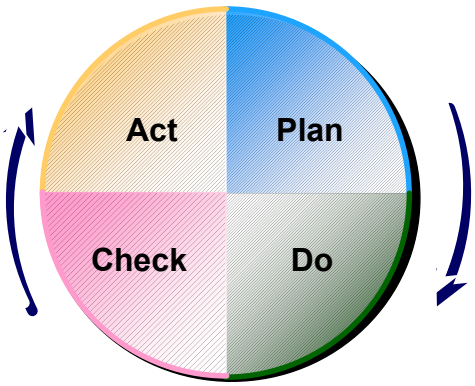
- Tester/ Developer Perceptions
  - Development output is real, testing output is intangible and not accountable
  - Testing is a very difficult effort (if done correctly)
  - Anyone can be a tester
  - Testers traditionally ignored until end of development
- Goal: Integrate testing and development
  - Testers are the "eyes to quality improvement" not the "code breakers"
  - Developers view testers as integral team players
  - Testers have save status as developers
  - Tester/ developer "real time" communication
  - Testers linked into project schedule
  - Test and development managers work closely together

A Continuous Quality Improvement Testing Methodology



- Apply Deming's manufacturing quality principles to software testing process
  - PDCA quality wheel
  - Constantly improve
    - **People**
    - **Process**
    - **Technology**
- Waterfall or RAD environments

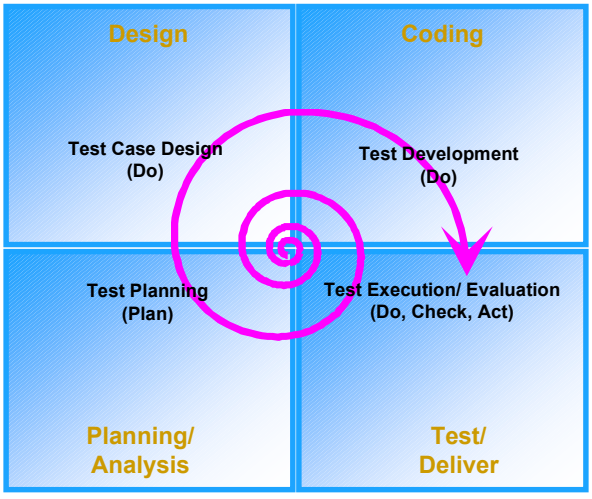
Applying the PDCA Quality Circle to Software Testing

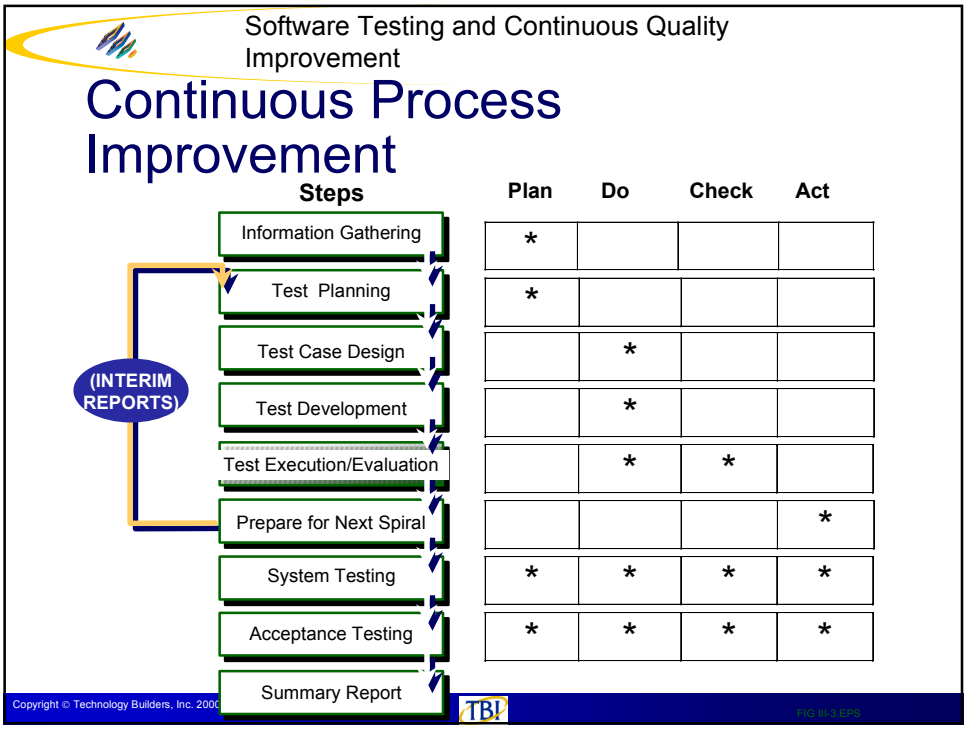


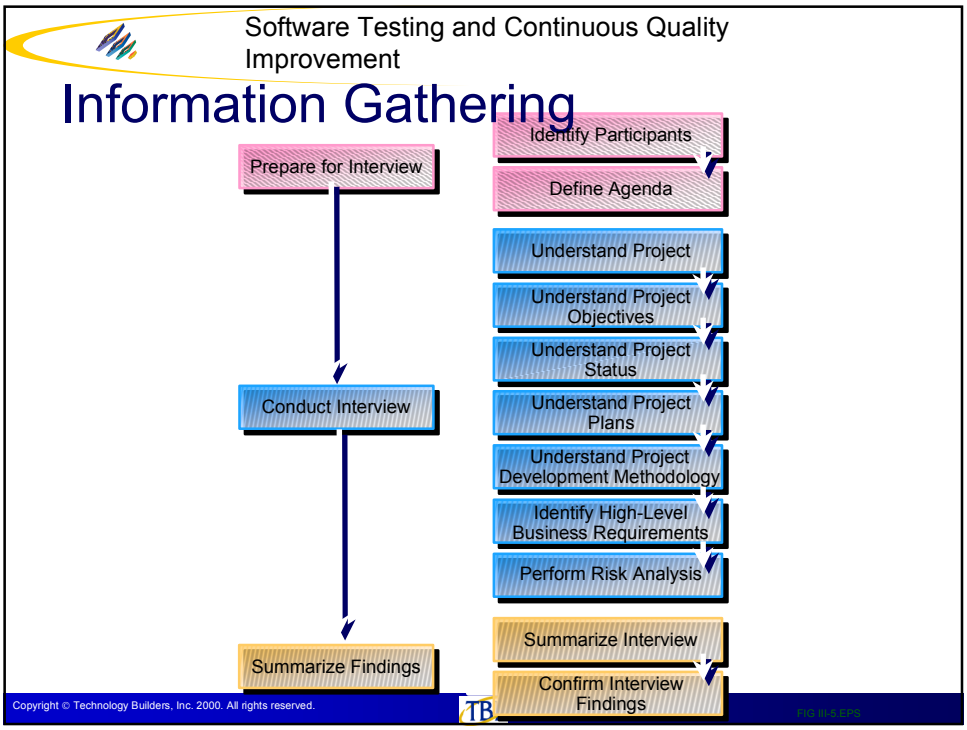
Components of PDCA Quality Wheel

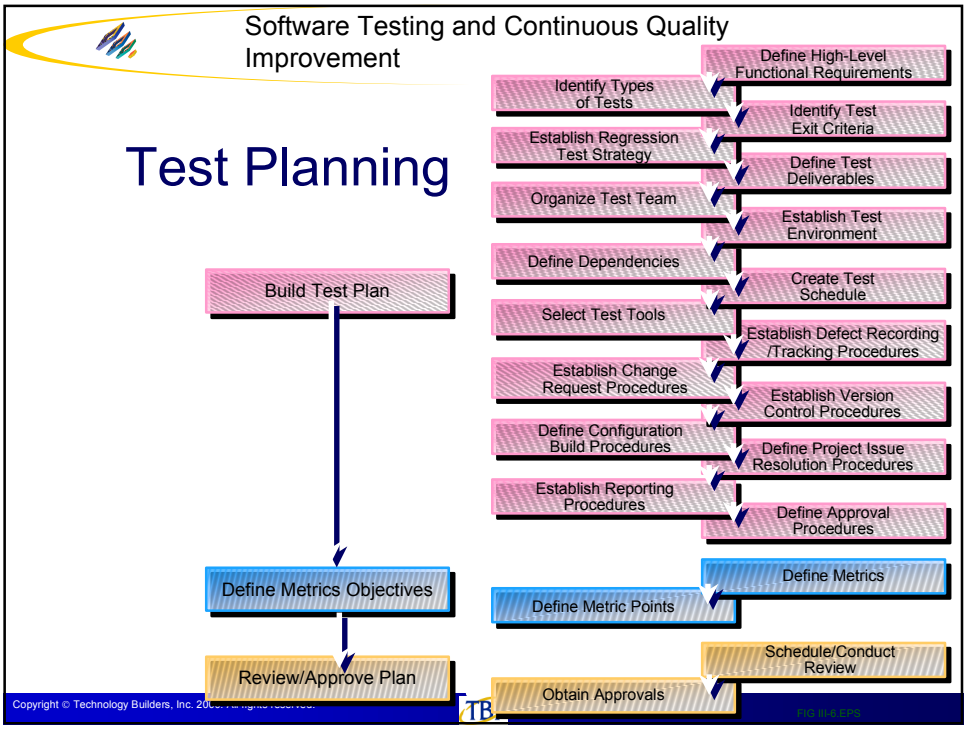
- Plan
  - Project interviewing and test planning, the basis of accomplishing testing
  - Define the metrics to control the people, process, technology
- Do
  - Test case design, test development, and test execution
- Check
  - Metric measurements and analysis
  - Evaluate work effort, resources, process, technology
- Act
  - Preparation for the next spiral
  - Refining the test plan and tests
  - Re-assessing test control procedures, test team

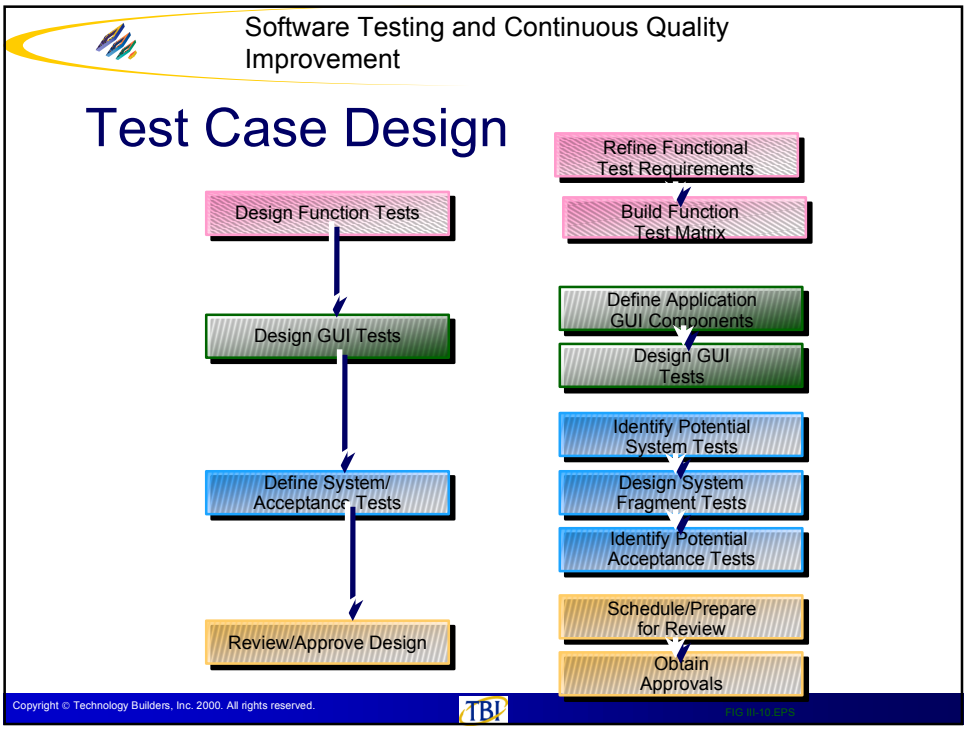
RAD/ Spiral Testing Methodology




Software Testing and Continuous Quality Improvement
Continuous Process Improvement
Steps
Plan
Do
Check
Act
Information Gathering
Test Planning
Test Case Design
Test Development
Test Execution/Evaluation
Prepare for Next Spiral
System Testing
Acceptance Testing
Summary Report
(INTERIM REPORTS)
Copyright © Technology Builders, Inc. 2000
TBI


Software Testing and Continuous Quality Improvement
Information Gathering
Prepare for Interview
Conduct Interview
Summarize Findings
Identify Participants
Define Agenda
Understand Project
Understand Project Objectives
Understand Project Status
Understand Project Plans
Understand Project Development Methodology
Identify High-Level Business Requirements
Perform Risk Analysis
Summarize Interview
Confirm Interview Findings
Copyright © Technology Builders, Inc. 2000. All rights reserved.
TB


Software Testing and Continuous Quality Improvement
Test Planning
Build Test Plan
Define Metrics Objectives
Review/Approve Plan
Define High-Level Functional Requirements
Identify Types of Tests
Identify Test Exit Criteria
Establish Regression Test Strategy
Define Test Deliverables
Organize Test Team
Establish Test Environment
Define Dependencies
Create Test Schedule
Select Test Tools
Establish Defect Recording /Tracking Procedures
Establish Change Request Procedures
Establish Version Control Procedures
Define Configuration Build Procedures
Define Project Issue Resolution Procedures
Establish Reporting Procedures
Define Approval Procedures
Define Metrics
Define Metric Points
Schedule/Conduct Review
Obtain Approvals
Copyright © Technology Builders, Inc.
TB


Software Testing and Continuous Quality Improvement
Test Case Design
Design Function Tests
Design GUI Tests
Define System/ Acceptance Tests
Review/Approve Design
Refine Functional Test Requirements
Build Function Test Matrix
Define Application GUI Components
Design GUI Tests
Identify Potential System Tests
Design System Fragment Tests
Identify Potential Acceptance Tests
Schedule/Prepare for Review
Obtain Approvals
Copyright © Technology Builders, Inc. 2000. All rights reserved.
TB

Test Development

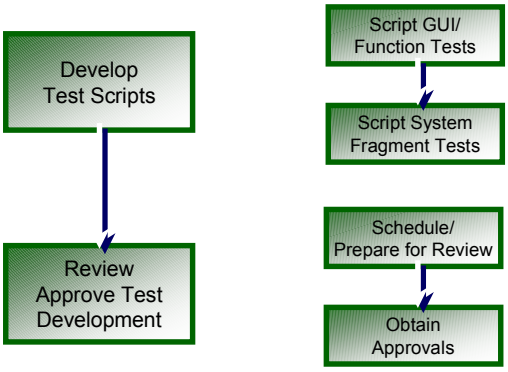


FIG III-14.EPS

Test Execution/Evaluation

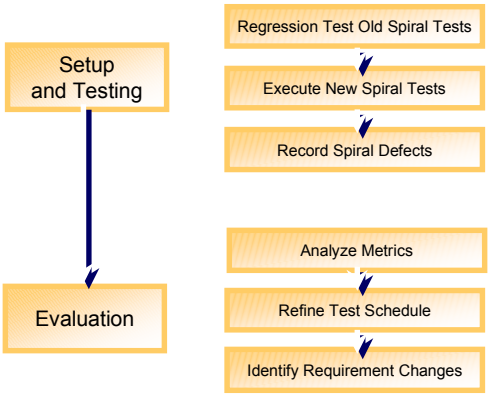


FIG III-16.EPS

Applying Statistics and Metrics

- Histograms (frequencies)
  - By priority
  - By severity
  - By tester
  - By environment
  - By time to fix
- Pareto Charts (ranked in ascending order)
  - By defect types
  - By defect source
  - By functional area defects
  - By program unit defects

Software Testing and Continuous Quality Improvement

Applying Statistics and Metrics (Con't)

- Run Trend Charts (over time)
  - Number of test cases designed, developed, executed
  - Number of functions tested over time
  - Number of test cases versus number of defects
  - Number of defects (per time period)
  - Cumulative number of defects ("burnout rate")
- Cause-and-Effect Diagrams (Ishikawa or fishbone)
  - Ex. Excessive size of of materials to be inspected leads to preparation rate that is too high
  - Ex. Preparation rate that is too high contributes to an excessive rate of inspection
  - Ex. Excessive rate of inspection causes fewer defects to be found

Prepare for the Next Spiral

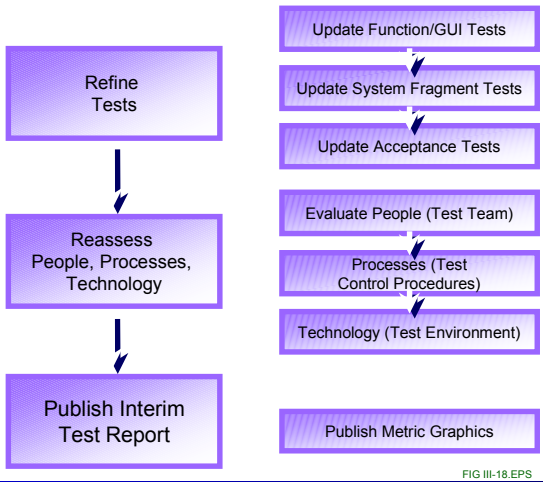
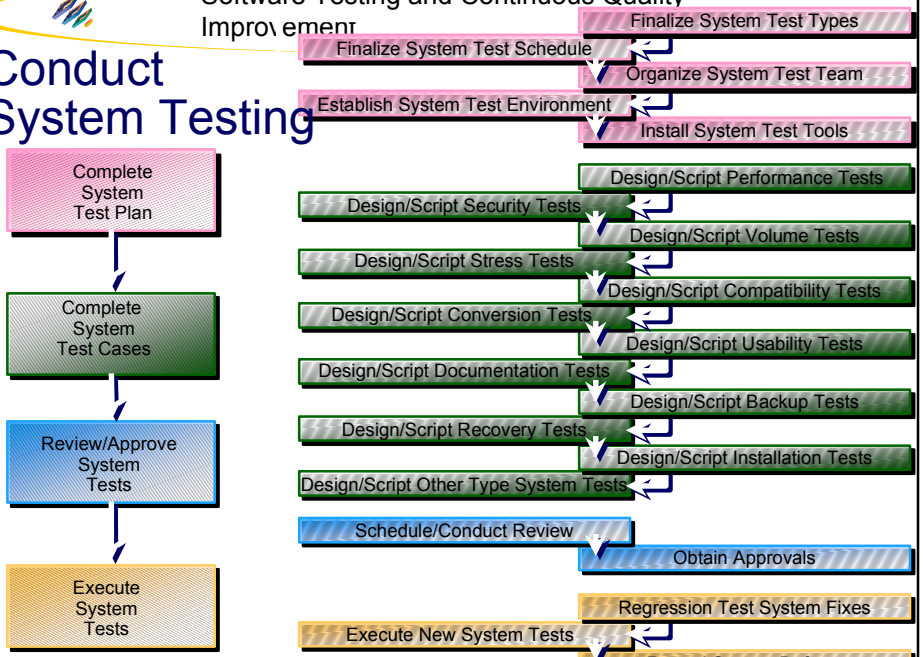
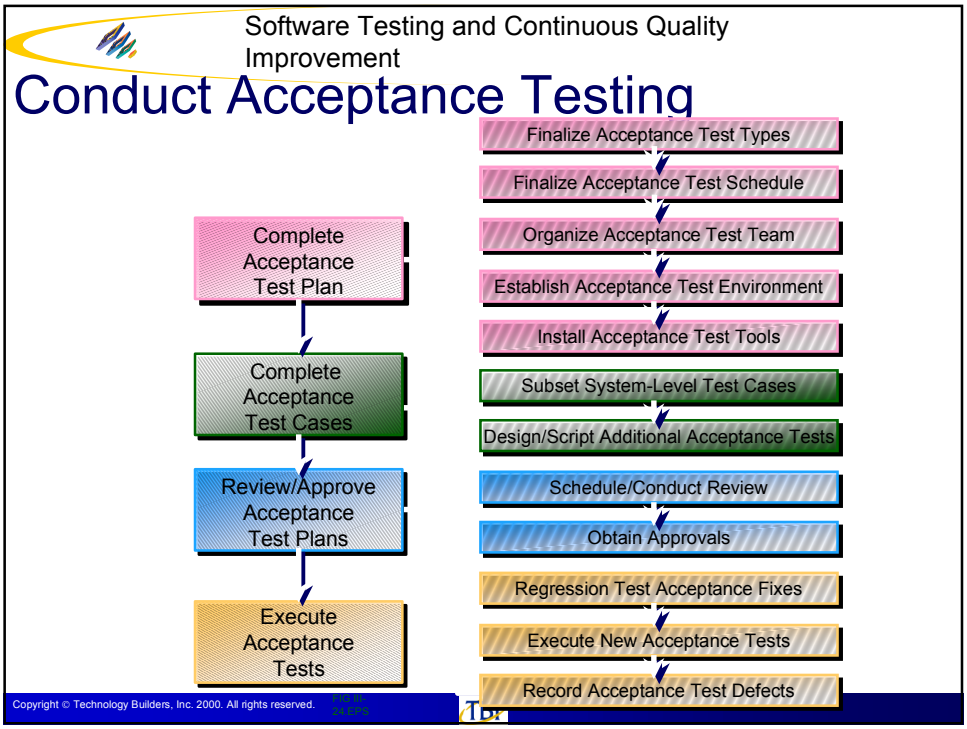
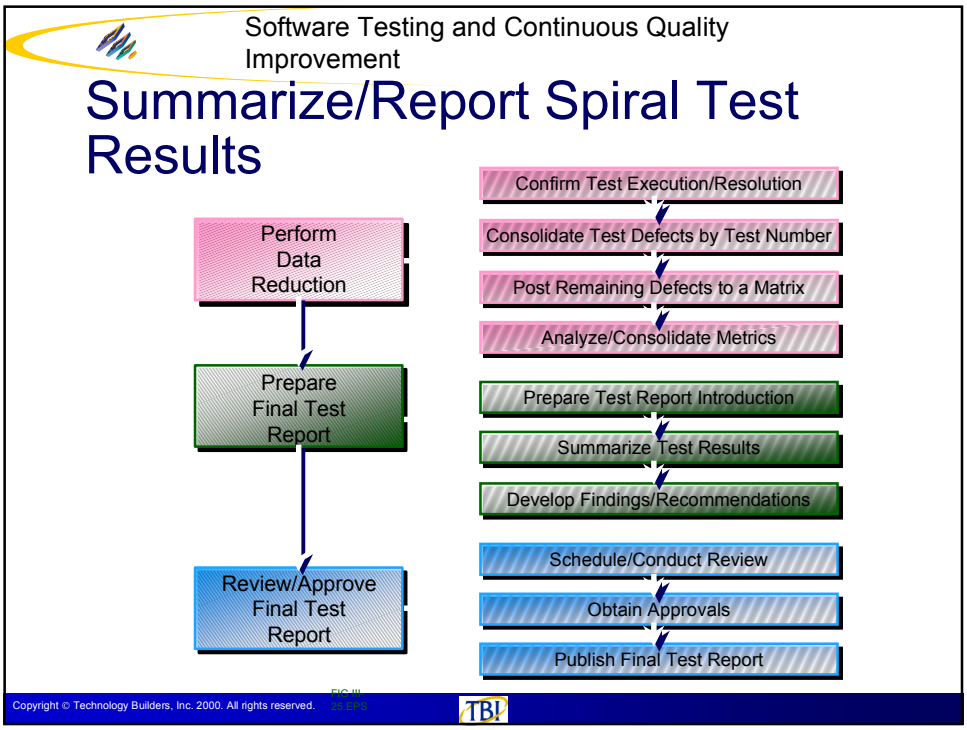


FIG III-18.EPS

Conduct System Testing




Software Testing and Continuous Quality Improvement
Conduct Acceptance Testing
Complete Acceptance Test Plan
Complete Acceptance Test Cases
Review/Approve Acceptance Test Plans
Execute Acceptance Tests
Finalize Acceptance Test Types
Finalize Acceptance Test Schedule
Organize Acceptance Test Team
Establish Acceptance Test Environment
Install Acceptance Test Tools
Subset System-Level Test Cases
Design/Script Additional Acceptance Tests
Schedule/Conduct Review
Obtain Approvals
Regression Test Acceptance Fixes
Execute New Acceptance Tests
Record Acceptance Test Defects
Copyright © Technology Builders, Inc. 2000. All rights reserved.


Software Testing and Continuous Quality Improvement
Summarize/Report Spiral Test Results
Perform Data Reduction
Prepare Final Test Report
Review/Approve Final Test Report
Confirm Test Execution/Resolution
Consolidate Test Defects by Test Number
Post Remaining Defects to a Matrix
Analyze/Consolidate Metrics
Prepare Test Report Introduction
Summarize Test Results
Develop Findings/Recommendations
Schedule/Conduct Review
Obtain Approvals
Publish Final Test Report
Copyright © Technology Builders, Inc. 2000. All rights reserved.

Reference

"Software Testing and Continuous Quality Improvement"
by William E. Lewis, Auerbach Publishers, 2000

ISBN O-8493-9833-9, Amazon.com

QWE2000 Session 11T

Mr. Richard Kasperowski & Mr. Spencer Marks
[USA]
(Altisimo Computing)

"Building Better Java Applications"

Key Points

- What is a "better" Java application?
- Best practices
- Methodologies

Presentation Abstract

While helping our clients build Java-based applications, we have learned practices and methodologies that result in successful projects. We will share the ingredients we consider most important in building better Java applications.

By "better" we mean that an application meets the sponsorÆs requirements, is flexible, and is delivered on time, all in an environment where change is constant and must be embraced rather than eschewed. The characteristics of a flexible application include ease of changing its feature set while under development, and extensibility and ease of maintenance after initial delivery. In addition, better Java applications encourage pleasant work environments.

Through our experience, we have recognized a specific set of best practices and methodologies that have helped us build better Java applications. Each best practice is a small scale, concrete procedure to follow while designing or implementing an application. In contrast, the methodologies are project-scope concepts that can be practiced in more than one way. In general, best practices are Java-specific, while methodologies may be language-neutral.

The majority of the presentation will describe each of the best practices and methodologies with which we have had success. Drawing from our experience, we will give a detailed description of each best practice and methodology, from both a theoretical and implementation-specific point of view.

About the Speaker

Richard Kasperowski is president of Altisimo Computing, a software development consulting firm based in Cambridge, Massachusetts. Richard has worked as tester, developer, manager, and consultant since 1988. He has a degree from Harvard University, is a member of the ACM, and usually cycles to his clientsÆ offices.

Spencer Marks is an independent software development consultant who has helped clients such as Apple Computer, Digital, Lotus, Symantec, and GTE design, code, and improve the quality of their products since 1989. For the past three years, he has focused exclusively on the Java language and related technologies such as XML. Spencer holds a bachelorÆs degree from Skidmore College and a masterÆs degree from Clark University.

Richard and Spencer have worked together on several projects since they were first introduced in 1988. Most recently they helped a major telephone company roll out a Java-based web application that allowed customers to view and pay their phone bills. Currently they are building the back-end and presentations layers to showcase an innovative new search engine technology.

BACK TO QWE2000 PROGRAM

Building Better Java Applications

Richard Kasperowski, Altisimo Computing
mailto:kasper@altisimo.com
http://www.altisimo.com

Spencer Marks, digisolutions
mailto:smarks@digisolutions.com
http://www.digisolutions.com

digisolutions Altisimo

Outline

- Introduction
- Best practices
- Methodologies
- Summary

digisolutions Altisimo

Introduction

- What is a "better" Java application?
- Definitions:
  - *Best practices*: small scale, concrete procedures
  - *Methodologies*: whole- project scope, multiple ways to implement

Best practice: Design patterns

- "Simple and elegant solutions to specific problems in object- oriented software design"
- Abstract Factory, Factory Method, Singleton, Adapter, Facade, Proxy, State, Strategy
- Gamma, *et al*; *Design Patterns: Elements of Reusable Object- Oriented Software*

Best practice: UML

- Unified Modeling Language: a visual language for modeling object oriented software systems
- Class diagrams
- State diagrams
- Together/J (http://www.togethersoft.com/)
- Fowler; *UML Distilled*

digisolutions Altisimo

Best practice: Jikes

- IBM- sponsored open source Java compiler
- Fast compile time
- Better error messages
- Better adherence to Java Language Spec.
- Cross- platform builds
- http://www10.software.ibm.com/developerworks/opensource/jikes/

digisolutions Altisimo

Best practice: Builds

- Use a single command to build the software product cleanly, completely, and quickly.
- Not interactive, enabling automated nightly builds
- Consistency across platforms and development tools
- `make`

digisolutions Altisimo

Best practice: Check in often

- Use a configuration management system, e.g. CVS
  - Learn how to use the tool.
- Small code changes
  - Easier to undo
  - Reduce impact on other coders
- Good tests encourage frequent check ins.
- Check in often to propagate changes quickly to the rest of the team.

digisolutions Altisimo

Best practice: Javadoc

- Standard way to document Java code and APIs
- Document all classes, attributes, operations
  - Class documentation: overview of how the class works and is intended to be used
  - Method documentation: explanation of how to use the method; not necessarily how the method works
- Does not replace in- line code comments

Best practice: Standard library

- Rich set of classes comes for free
- Don' t reinvent the wheel, e.g.:
  - Use `Properties` for run- time parameterization
  - Many design patterns, e.g. `Iterator`, are already implemented.
  - `Collections` are handy.

Best practice: Code to interfaces

- Interfaces vs. classes
- Isolate subsystems from code changes
  - Easy to change implementation without affecting client code
  - Implementation can be specified at run- time, instead of at compile time.

Best practice: Java idioms

- Example: Use an `Iterator` or `Enumeration` instead of a `for` loop to iterate over the elements of a `Collection`.
- Larman, Guthrie; *Java 2 Performance and Idiom Guide*

Methodology: Write tests first

- Extreme Programming
- Easier to gauge progress
  - Specify software product's behavior.
  - Write tests that would validate that behavior.
  - Write product code until it passes tests.
- http://www.xprogramming.com/

Methodology: Refactor often

- Don't copy and paste code
  - Code says to programmer, "Please refactor me."
- "Write tests first" makes refactoring safe and easy.
  - Programmer says to code, "OK, no problem."
- Fowler, *et al*; *Refactoring : Improving the Design of Existing Code*

Methodology: Frequent build and test

- Automated build and test
- Weekly? Nightly? Hourly? At each check in?
- Catch broken code changes as soon as possible
- Ensures there' s always a build you can ship.

digisolutions Altisimo

Methodology: Pair programming

- XP
- Teams of two people
  - Real- time peer review
  - Each person has a specific role.
- Also pair designing, pair testing

digisolutions Altisimo

Methodology: Use cases, metaphors

- Use cases (UML): "a typical interaction between a user and a computer system"
- Metaphors (XP): "a common ' system of names' and a common system description that guides development and communication"

digisolutions Altisimo

Methodology: Requirements change

- Expect requirements to change; it is normal.
  - Don' t build a monolithic software specification.
  - Build a small spec. that evolves as requirements change.
- Effectively deal with changes
  - Each changed requirement is an opportunity for refactoring and other improvements.

digisolutions Altisimo

Methodology: Keep it simple

- Do the simplest thing that works.
- Implement desired features quickly.
- Don't add unspecified features that "might be nice someday."
- Balance short- term and long- term goals.

Methodology: Efficiency

- Know when to worry about efficiency and code optimization.
  - Usually after all features have been implemented
  - Don't worry about inefficient implementations in the standard library until later.
- Avoid inefficient algorithms in general.

Methodology: Small achievable tasks

- Break tasks into small achievable goals.
- Short feedback loops
- Easier to gauge progress
- Get new code changes to the rest of the team quickly.
- Easier to deal with changed requirements

digisolutions Altisimo

Methodology: Separate test code

- ... from product code in a way that is effective for the project.
- Examples:
  - Tests in same directory as classes they test
  - Tests in subdirectory as classes they test
  - Tests in separate, parallel directory tree

digisolutions Altisimo

Summary

- Best practices
  - Design patterns, UML, Jikes, easy builds, frequent check- ins, Javadoc, standard library, interfaces, idioms
- Methodologies
  - Write tests first, refactor, frequent build and test, pair programming, use cases / metaphors, requirements change, KISS, efficiency, small tasks, separate test code

digisolutions Altisimo

Richard Kasperowski, Altisimo Computing

mailto:kasper@altisimo.com

http://www.altisimo.com

Spencer Marks, digisolutions

mailto:smarks@digisolutions.com

http://www.digisolutions.com

digisolutions Altisimo

QWE2000 Session 11A

Dr. Nigel Bevan
(Serco Usability Services, UK) & Mr. Itzhak Bogomolni
(Israel Aircraft Industries, Israel)

"Incorporating User Quality Requirements In The Software Development Process"

Key Points

- Usability
- User centred design
- User requirements

Presentation Abstract

Why are methods for incorporating user quality requirements and improving usability not more widely adopted? There is compelling evidence for the cost benefits of user centred design, development time can be reduced, sales increased, the productivity of users improved, and support and maintenance costs reduced. But although the process is supported by international standards it has been difficult to integrate into mainstream software development.

The EU INUSE and RESPECT projects developed a complete set of methods for incorporating user quality requirements in development, for testing usability and for assessing the usability maturity of organisations.

Our experience of using the methods will be described in the paper.

After application of the methods over a period of 12 months, LAHAV assessed the benefits. The conclusions were very positive. The paper will include the results of the second process improvement assessment.

About the Speaker

Nigel Bevan is Research Manager at Serco Usability Services. He manages the EU T RUMP project described in this paper. He has developed and applied methods for usability and user centred design, and has been editor of several international standards for usability and software quality.

Itzhak Bogomolni leads software process improvement at the LAHAV division of IAI and has spent the last eight years working on process improvement activities. His

previous experience was in development of embedded real time systems.

Incorporating user quality requirements in the software development process

Nigel Bevan, Serco Usability Services
Itzhak Bogomolni, Israel Aircraft Industries
nbevan@usability.serco.com
ibogomolni@lahav.iai.co.il
www.usability.serco.com/trump

Topics

- Usability and user requirements
- Conventional approach to usability
- Means of achieving Quality in Use
- TRUMP approach and methods
- IAI experience with TRUMP methods
- Trial Application conclusions

How Usable are current systems ?

- UK Passport Office
  - New software for issuing passports took operators twice as long
  - Caused delays of up to 3 months in obtaining a passport
  - Huge cost of additional clerical staff
- E-commerce web sites
  - User success in purchasing ranges from 25%-42%

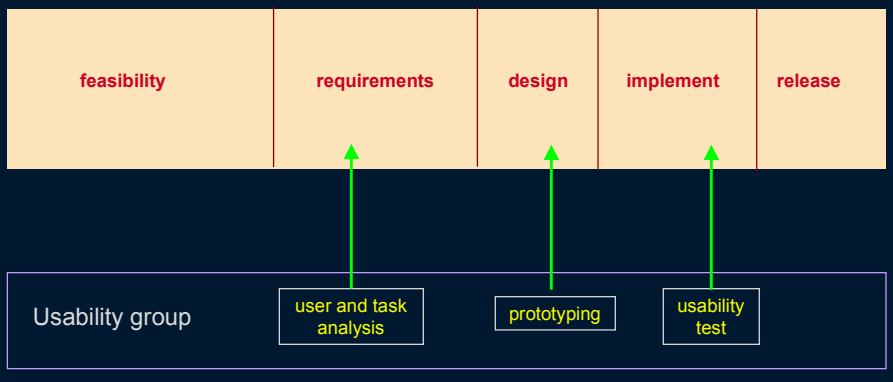
What is the Quality of the User Requirements ?

- Standish Group found that
  - 51% of projects failed
  - 31% were partially successful
- Main causes were poor user requirements:
  - **13.1% Incomplete requirements**
  - **12.4% Lack of user involvement**
  - 10.6% Inadequate resources
  - **9.9% Unrealistic user expectations**
  - 9.3% Lack of management support
  - **8.7% Requirements keep changing**
  - 8.1% Inadequate planning
  - **7.5% System no longer needed**

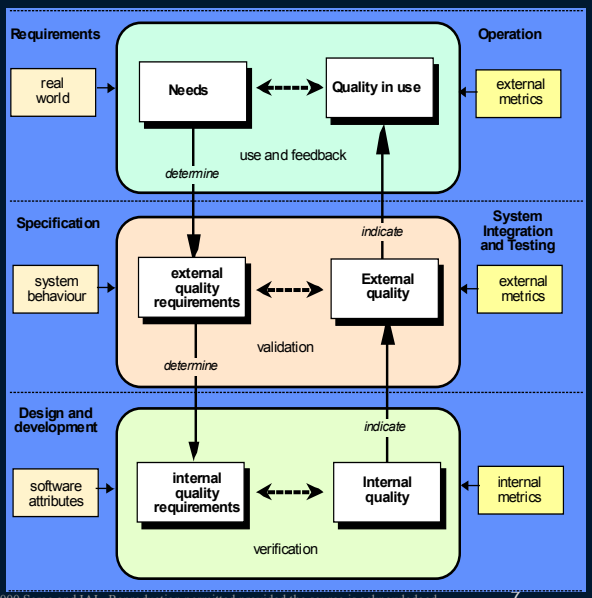
Who is responsible for system usability and users requirements ?

- Who (if anyone) in your organization is responsible for System usability (what organizational function) ?
- Does your organization specify non-functional User Requirements ?
- How can the user requirements be improved?
- How can the systems usability be improved?

Conventional approach to usability

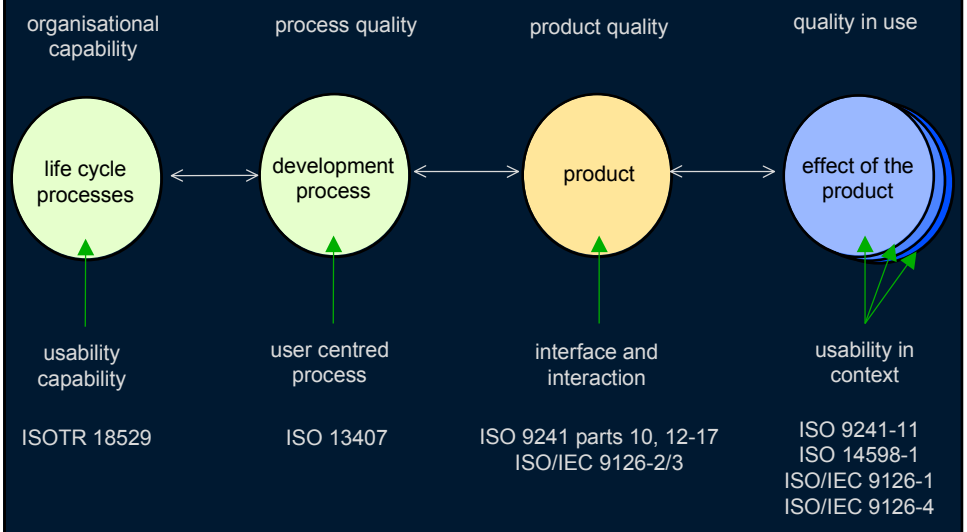


The Need for Quality in Use



ISO/IEC 14598-1 Software Product Evaluation - General Overview

Means of achieving Quality in Use



ISOTR 18529

ISO 13407

ISO 9241 parts 10, 12-17
ISO/IEC 9126-2/3

ISO 9241-11
ISO 14598-1
ISO/IEC 9126-1
ISO/IEC 9126-4

TRUMP: Trial Usability Maturity Process

- EU-funded trial application of user-centred design methods developed in previous research projects (INUSE and RESPECT)
- Serco: apply the methods
  - Lloyds Register: Usability Maturity Assessment
- Inland Revenue/EDS - IT for 60,000 staff
  - RAD methodology
- Israel Aircraft Industries - aerospace systems
  - traditional methodology

LAHAV Background

- A Division of Israel Aircraft Industries
- Expertise in Military Aircraft "Avionics Upgrade Programs"
- Customers Worldwide
- Avionics directorate - 100+ developers (Pilots, System, Software, Facilities)
- User needs addressed by group of Pilots

TRUMP at LAHAV

- LAHAV TRUMP objectives
  - Improve Operational Requirements Definition and Evaluation Process
  - Increase Usability of LAHAV Products
  - Increase Customer Satisfaction from LAHAV Products
- TRUMP's Trial Application : Mission Planning Center (MPC)
  - The system is used to plan an airborne mission - Standard NT Interface
  - The mission is then loaded onto a cartridge that is taken by the pilot to the aircraft.
  - In the aircraft the pilot loads the data into the aircraft's main mission computer

TRUMP Approach

Plan Process	Specify Context of Use	Specify Requirements	Design Solutions	Evaluate against Requirements
System lifecycle				
feasibility		requirements	design	implement / release
1. Stake-holder meeting	2. Context of use 3. Scenarios	4. Usability requirements 5. Evaluate existing system	6. Prototyping 7. Style guide	8. Evaluation 9. Usability testing 10. Collect feedback

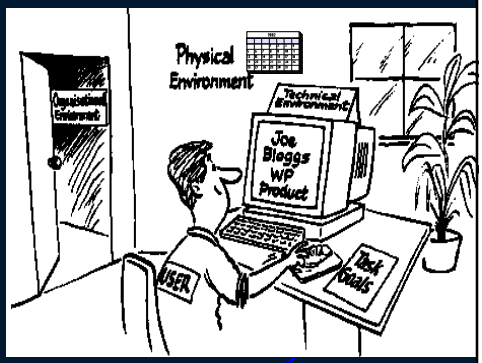
www.usability.serco.com/trump

1 - Define Goals and Stakeholders

- Method definition
  - A half-day meeting to identify and agree on the role of usability, broadly identifying the intended context of use and usability goals, and how these relate to the business objectives and success criteria for the system
- What we did
  - New goals and objectives were defined.
  - New intended user and stakeholders were identified.
  - Projects scope and objectives were discussed and issues identified.
- Method evaluation
  - Identified new goals, users and stakeholders
  - Better understood scope and objectives

2- Context of Use

- The usability of a product is affected not only by the features of the product itself but also by its Context of Use
- Context is the characteristics of:
  - the users of the product
  - the tasks they carry out
  - the technical, organisational and physical environment in which the product is used
  - the date and time when the product is being used



2 - Context of Use (Cont.)

- Method definition
  - A half-day workshop to collect and agree detailed information about the intended users, their tasks, and the technical and environmental constraints.
- What we did
  - The context of use was defined.
  - The user's skills, tasks and the working environment were defined.
- Method evaluation
  - The checklist is too long and should be customised for every system being developed.
  - An experienced facilitator is very important to the success of the technique.

3 - Produce Scenarios

- Method definition
  - A half-day workshop to document examples of how users are expected carry out key tasks in a specified context, to provide an input to design and a basis for subsequent usability testing.
- What we did
  - Produced two detailed operational scenarios of how the MPC might be used
- Method evaluation
  - The few operational scenarios required for MPC are obvious for Pilots.
  - It was concluded that this technique was not so relevant for MPC.
  - Another Trial will be conducted on Avionics project to evaluate the technique relevance to LAHAV
  - The contribution for MPC system was low.

4 - Task Analysis

- Method definition
  - Workshop: everyone produces sticky notes for all imagined functions, which are sorted into logical groups on the wall. The notes are arranged into a hierarchy of functions to support common user tasks.
- What we did
  - The technique has been customised in advance.
  - Functions and features were added and functional hierarchy has been changed significantly and agreed upon.
  - System architecture has been modified accordingly.
- Method evaluation
  - The method had a great impact on the MPC look and feel, it's S/W requirements and architecture

5 - Evaluate Usability of Existing System

- Method definition
  - Evaluate an earlier version or competitor system to identify usability problems and obtain measures of usability as an input to usability requirements.
- What we did
  - Existing MPC has been used by 4 pilots to prepare a mission (The MPC main task).
  - About 50 usability problems have been identified.
  - SUMI questionnaires have been filled and results evaluated.
- Method evaluation
  - The technique was very productive though has been applied in a semi-formal way.
  - More formal trial should be considered.

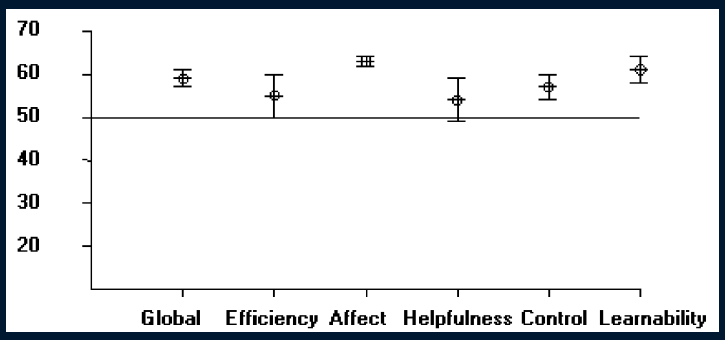
6 - Quality in Use Goals and Usability Requirements

- Method definition
  - A half-day workshop to establish usability requirements for the user groups and tasks identified in the context of use analysis and in the scenarios.
- What we did
  - Usability time goals were defined.
  - A list of probable errors was created.
- Method evaluation
  - The technique is not well defined and not fully understood at LAHAV.
  - LAHAV realises the need for the technique and it's potential
  - More work is needed to better define it.

7 - Paper Mockup Prototyping

- Method definition
  - Evaluation by users of quick low fidelity prototypes(using paper or other materials) to clarify requirements and enable draft interaction designs and screen designs to be rapidly simulated and tested.
- What we did
  - The Screens Printouts were posted on the wall and provided the "Big Picture".The overheads of the screens were extensively used.
  - Four major GUI sections were conceived.
  - Sections order was modified according to logical planning flow.
  - Interface comments modified screens' layout.
  - An item manipulation technique was agreed upon
  - A list of 23 usability comments was created
- Method evaluation
  - Very fruitful and productive discussions.

SUMI



Usability test of new MPC prototype

8 - Formal and Informal Usability Testing

- Method definition
  - Informal usability testing with 3-5 representative users carrying out key tasks to provide rapid feedback on the usability of prototypes.
  - Formal usability testing with 8 representatives of a user group carrying out key tasks to identify any remaining usability problems and evaluate whether usability objectives have been achieved.
- What we did
  - Informal Computer Prototype Usability Testing
    - The pilot's interaction with the computer was projected on a wall. The evaluators and the developer sat in the same room observing the interaction. The pilot was encouraged to think aloud.
    - 97 problems were identified after six pilots had tried the system.
  - Formal Usability Testing against Requirements
    - The system was tested against timing requirements defined for two typical tasks. Eight pilots had 2-hours frontal training followed by 2-hours hands-on practice. Each pilot performed two tasks according to a written tasks definition. Task duration's were recorded and scored well. 54 "points of detail"comments were identified.
- Method evaluation
  - Both methods were very valuable and almost all the problems were subsequently fixed.

Trial Application Bottom Line

- Positive Feedback from Participants
- A Definite Improvement in the Development Process
- Very Cost Effective and Low Cost
- Mostly Intuitive however tailoring sometimes required
- Expert Guidance needed in few techniques
- LAHAV decided to incorporate TRUMP techniques in it's standard development process

Summary

- Low cost simple methods for improving systems usability are here
- The methods can be easily integrated into any software development process
- The responsibility for quality in use should be transferred to development organization
- The Methods and supporting means are available on the Web
- No more Excuses

Incorporating user quality requirements in the software development process

Nigel Bevan[*] and Itzhak Bogomolni[+]

[*]Serco Usability Services, 4 Sandy Lane, Teddington, Middx, TW11 0DU, UK
[+] Israel Aircraft Industries Ltd, Ben Gurion International Airport, 70100, Israel

nbevan@usability.serco.com, ibogomolni@lahav.iai.co.il

www.usability.serco.com/trump

Abstract

The business worth of a computer system is a function of its quality in use – the extent to which it is fitted for its purpose. ISO/IEC 14598-1 (Evaluation of Software Products) places quality in use as the overall goal for software development. The term quality in use recognises that software does not exist in isolation, but must fit with a socio-technological work environment if it is to work in practice.

A major obstacle to achieving the goal of consistently usable systems is a lack of guidance on integrating the various techniques available to achieve the required process. Quality in use is increasingly recognised in industry as a primary goal in developing business systems and IT products.

The objective of the EU TRUMP project was to directly increase the quality of products and systems by assisting in the integration of usability methods into the existing systems development processes, and by the promotion of usability awareness into the culture of the organisations.

The methods were applied in trial projects over a 12-month period. In both cases the results were judged to be highly beneficial and cost effective, and the selected methods are now being formally incorporated into the organisations' development processes.

1. The need for quality in use

The purpose of designing an interactive system is to meet the needs of users: to provide quality in use (Bevan, 1999). The internal software attributes will determine the quality of a software product in use in a particular context. Software quality attributes are the cause, quality in use the effect (see Figure 1, from ISO/IEC 14598-1). Quality in use is (or at least should be) the objective, software product quality is the means of achieving it.

The users' needs can be expressed as a set of requirements for the behaviour of the product in use (for a software product, the behaviour of the software when it is executed). These requirements will depend on the characteristics of each part of the overall system including hardware, software and users.

The requirements should be expressed as metrics that can be measured when the system is used in its intended context. The required system characteristics could be minimum values for the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in specified environments.

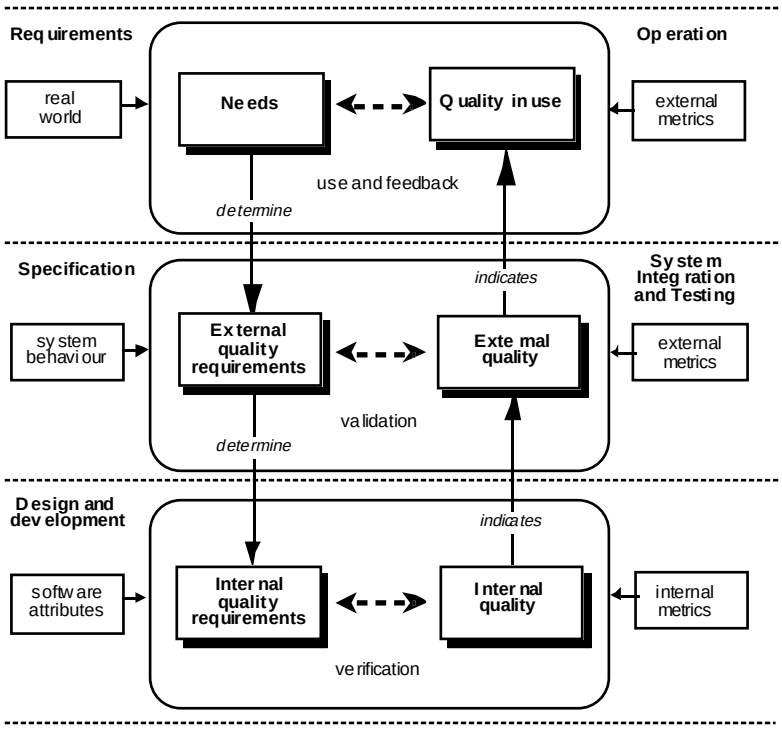


Figure 1. Quality in the software lifecycle

2. Means of achieving quality in use

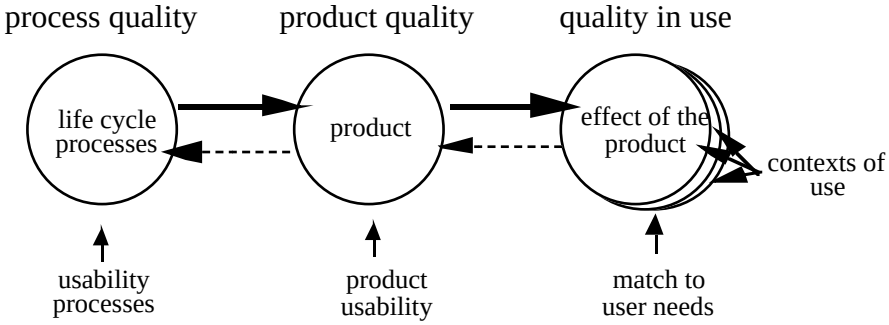


Figure. 2. Approaches to achieving quality in use

The TRUMP project combined three complementary approaches to improving the quality of a product from a user perspective (Figure 2):

- Improve the quality of the software development processes, by incorporating user-centred activities derived from ISO 13407 and the Usability Maturity Model in ISO TR 18529.
- Improve the quality of the software: by improving the quality of the user interface.
- Improve the quality in use: by ensuring that the software meets the needs of the user for effectiveness, productivity and satisfaction in use.

The quality of the software development process can be improved through use of ISO 13407 and ISO TR 18529 that define user centred activities.

2.1 User centred design process: ISO 13407

ISO 13407 provides guidance on achieving quality in use by incorporating user centred design activities throughout the life cycle of interactive computer-based systems. It describes user centred design as a multi-disciplinary activity, which incorporates human factors and ergonomics knowledge and techniques with the objective of enhancing effectiveness and productivity, improving human working conditions, and counteracting the possible adverse effects of use on human health, safety and performance.

There are four user centred design activities that need to start at the earliest stages of a project. These are to:

- understand and specify the context of use
- specify the user and organisational requirements
- produce design solutions
- evaluate designs against requirements.

The iterative nature of these activities is illustrated in Figure 3. The process involves iterating until the objectives are satisfied.

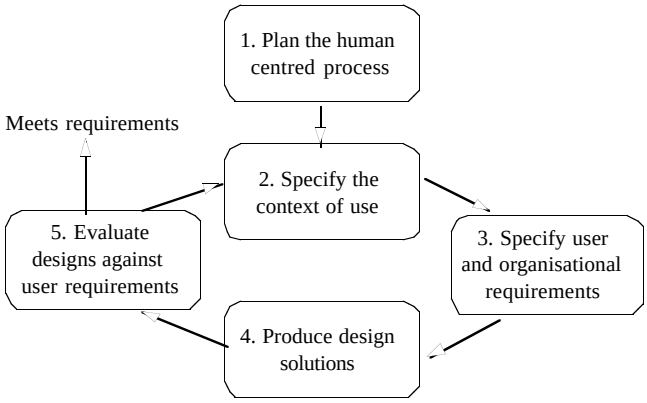


Figure 3 - The interdependence of user centred design activities

The sequence in which these are performed and the level of effort and detail that is appropriate varies depending on the design environment and the stage of the design process.

2.2 Human-centred lifecycle process descriptions: ISO TR 18529

INUSE developed a structured and formalised definition of the human-centred processes described in ISO 13407 (Earthy 1998). An improved version has subsequently been published as ISO TR 18529. It is intended to make the contents of ISO 13407 accessible to software processes assessment and improvement specialists and to those familiar with or involved in process modelling. It can be used in the specification, assessment and improvement of the human-centred processes in system development and operation.

The model consists of seven sets of base practices (Figure 4). These base practices describe what has to be done in order to represent and include the users of a system during the lifecycle. The model uses the format common to process assessment models. These models describe the processes that ought to be performed by an organisation to achieve defined technical goals. The processes in this model are described in the format defined in ISO 15504 *Software process assessment*. Although the primary use of a process assessment model is for the measurement of how well an organisation carries out the processes covered by the model, such models can also be used as a description of what is required in order to design and develop effective organisational and project processes.

HCD.1	**Ensure HCD content in system strategy**
HCD.1.1	Represent stakeholders
HCD.1.2	Collect market intelligence
HCD.1.3	Define and plan system strategy
HCD.1.4	Collect market feedback
HCD.1.5	Analyse trends in users
HCD.2	**Plan and manage the HCD process**
HCD.2.1	Consult stakeholders
HCD.2.2	Identify and plan user involvement
HCD.2.3	Select human-centred methods and techniques
HCD.2.4	Ensure a human-centred approach within the project team
HCD.2.5	Plan human-centred design activities
HCD.2.6	Manage human-centred activities
HCD.2.7	Champion human-centred approach
HCD.2.8	Provide support for human-centred design
HCD.3	**Specify the stakeholder and organisational requirements**
HCD.3.1	Clarify and document system goals
HCD.3.2	Analyse stakeholders
HCD.3.3	Assess risk to stakeholders
HCD.3.4	Define the use of the system
HCD.3.5	Generate the stakeholder and organisational requirements
HCD.3.6	Set quality in use objectives
HCD.4	**Understand and specify the context of use**
HCD.4.1	Identify and document user's tasks
HCD.4.2	Identify and document significant user attributes
HCD.4.3	Identify and document organisational environment
HCD.4.4	Identify and document technical environment
HCD.4.5	Identify and document physical environment
HCD.5	**Produce design solutions**
HCD.5.1	Allocate functions
HCD.5.2	Produce composite task model
HCD.5.3	Explore system design
HCD.5.4	Use existing knowledge to develop design solutions
HCD.5.5	Specify system and use
HCD.5.6	Develop prototypes
HCD.5.7	Develop user training
HCD.5.8	Develop user support
HCD.6	**Evaluate designs against requirements**
HCD.6.1	Specify and validate context of evaluation
HCD.6.2	Evaluate early prototypes in order to define the requirements for the system
HCD.6.3	Evaluate prototypes in order to improve the design
HCD.6.4	Evaluate the system in order to check that the stakeholder and organisational requirements have been met
HCD.6.5	Evaluate the system in order to check that the required practice has been followed
HCD.6.6	Evaluate the system in use in order to ensure that it continues to meet organisational and user needs
HCD.7	**Introduce and operate the system**
HCD.7.1	Management of change
HCD.7.2	Determine impact on organisation and stakeholders
HCD.7.3	Customisation and local design
HCD.7.4	Deliver user training
HCD.7.5	Support users in planned activities
HCD.7.6	Ensure conformance to workplace ergonomic legislation

Figure 4. Human-centred design processes and their base practices

3. Benefits of user centred design

Given these international standards for user-centred design, why is it not more widely adopted? There is compelling evidence for the cost benefits (Bias and Mayhew, 1994), development time can be reduced, sales increased, the productivity of users improved, and support and maintenance costs reduced.

Development Usability engineering can reduce the time and cost of development efforts through early definition of user goals and usability objectives, and by identification and resolution of usability issues. Keil and Carmel (1995).

Sales There is increasing market demand for products that are easy to use.

Use Companies that purchase or produce usable systems for their employees can benefit from:

- *Increased effectiveness*. Avoiding inconsistencies, ambiguities or other interface design faults will increase effectiveness by reducing user error.
- *Increased efficiency*. A system incorporating a user interface designed to meet the needs of the task will allow the user to be more productive.
- *Improved satisfaction*: User acceptance is particularly important for applications like web sites where usage is discretionary.

Support and Maintenance A well-designed system designed with a focus on the end-user can reinforce learning, thus reducing training time and effort and support costs .

According to IBM (1999) “It makes business effective. It makes business efficient. It makes business sense”.

Reasons for the limited take up include the perceived high costs and the specialist skills required. The objective in the TRUMP project was to select a set of methods that are both cost-effective and easy to learn and to use.

4. TRUMP methods

The user centred design techniques recommended by TRUMP were selected to be simple to plan and apply, and easy to learn by development teams. Figure 5 shows how each of the recommended methods relates to the lifecycle stages and the processes described in ISO 13407.

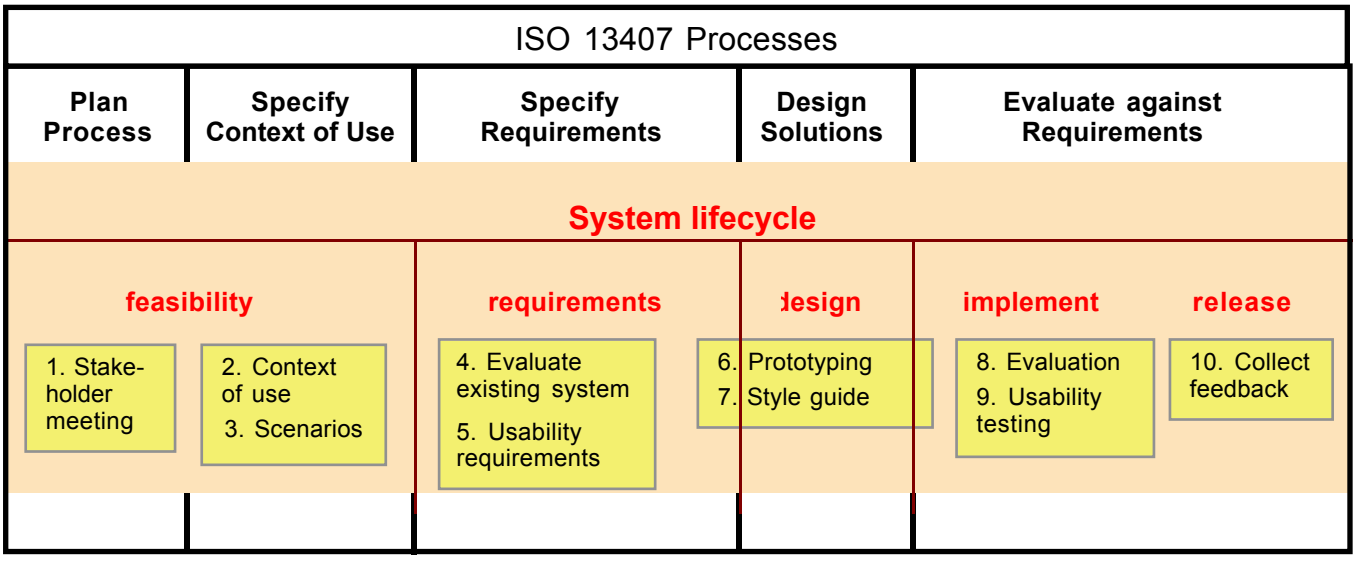


Figure 5. TRUMP Methodology

Each of the methods in Figure 5 is described below.

1. Stakeholder meeting

A half-day meeting to identify and agree on the role of usability, broadly identifying the intended context of use and usability goals, and how these relate to the business objectives and success criteria for the system.

2. Context of use

A half-day workshop to collect and agree detailed information about the intended users, their tasks, and the technical and environmental constraints.

3. Scenarios of use

A half day workshop to document examples of how users are expected carry out key tasks in a specified contexts, to provide an input to design and a basis for subsequent usability testing.

4. Evaluate an existing system

Evaluate an earlier version or competitor system to identify usability problems and obtain measures of usability as an input to usability requirements.

5. Usability requirements

A half-day workshop to establish usability requirements for the user groups and tasks identified in the context of use analysis and in the scenarios.

6. Paper prototyping

Evaluation by users of quick low fidelity prototypes (using paper or other materials) to clarify requirements and enable draft interaction designs and screen designs to be rapidly simulated and tested.

7. Style guide

Identify, document and adhere to industry, corporate or project conventions for screen and page design.

8. Evaluation of machine prototypes

Informal usability testing with 3-5 representative users carrying out key tasks to provide rapid feedback on the usability of prototypes.

9. Usability testing

Formal usability testing with 8 representatives of a user group carrying out key tasks to identify any remaining usability problems and evaluate whether usability objectives have been achieved.

10. Collect feedback from users

Collect information from sources such as usability surveys, help lines and support services to identify any problems that should be fixed in future versions.

5. TRUMP trials

TRUMP applied these methods in two contrasting environments: the Inland Revenue (IR) in the UK, which provides data processing to support 60,000 staff in more than 600 local offices; and the LAHAV division of Israel Aircraft Industries (IAI) in Israel, which has a group of about 100 people developing aircraft avionics. IAI uses a well-established development methodology, but their process for specifying operational requirements is not supported by any specific methods and

techniques. Inland Revenue employs a well-defined rapid application design (RAD) methodology in conjunction with its IT partner EDS.

This paper describes the experience at IAI. More information about both trials can be found in Bevan and Bogomolni (2000) and the TRUMP web site www.usability.serco.com/trump.

6. Trial at IAI

The Avionics directorate at Lahav division of Israel Aircraft Industries is responsible for providing modern avionics solutions and support products for modernised aircraft. It is a relatively small entity about 100 people.

The avionics upgrade projects follow a well established mature engineering process starting with concept definition through requirements, design, software development, system integration to flight testing by the customer.

User needs are addressed by a group of IAI pilots who represent the customer/user and define the operational requirements. Their work is based on their operational experience and previous projects, but is not supported by any specific methods and techniques.

Lahav is part of IAI-wide process improvement program that started at 1992. The program initially focused on software, adopted SEI Capability Maturity Model as a map for improvement. In following years process improvement assets and a support infrastructure was created and contributed to successful introduction of processes, methods and technologies.

LAHAV joined the TRUMP project with the objective of evaluating the impact of applying user-centred methods on a typical project. Lahav had the following business objectives:

- Improve the operational requirements definition and evaluation process
- Increase usability of LAHAV products
- Increase customer satisfaction from LAHAV products

At a more detailed level we wanted to:

- Assess the techniques' contribution to usefulness of the developed product.
- Understand how these techniques can be integrated into IAI development process.
- Measure the costs of applying the techniques.
- Evaluate developers' and managers' readiness to practice these techniques and the degree of their satisfaction from the process and their results.

We learned from our process improvement experience that the last objective is especially important for successful introduction of new methods.

6.1 Selection of methods

We selected the development of a new Mission Planning Centre (MPC) using the Windows NT Interface as a trial project. An MPC enables a pilot to plan an airborne mission that is then loaded onto a cartridge and taken by the pilot to the aircraft. In the aircraft the pilot loads the data into the aircraft’s main mission computer

We started with a one-day informal workshop-style assessment against the Usability Maturity Model (UMM) performed by Serco. A series of interviews with developers and managers were held throughout the day to rate the extent to which each base practice was carried out.

Then we selected which methods to use for the trial. The selection was based on:

- The areas for improvement identified in the UMM assessment

- The specifics of MPC project
- Ease of integration with the IAI development process
- Our intuition relating the potential value of each technique

6.2 IAI Experience with the methods

6.2.1 Stakeholder meeting

We used to conduct a project initiation meeting involving a project development management and technical staff. User related (Operational requirements) were separately defined and discussed by a specialised Pilot's group. Conducting a Stakeholders meeting allowed to identify previously unforeseen users and stakeholders, better understand the project scope and objectives, define the success factors and identify some different interpretations for follow-up discussions and resolution. Involvement of senior managers and marketing personnel contributed for identification of some strategic issues.

6.2.2 Analyse context of use

We never used this method before. The facilitator guided us through a long checklist covering many aspects of the user's skills, tasks and the MPC working environment. Many terms were not familiar to us and required explanation. Most of the data captured was not new to the participants due to their good familiarity with users environment. Some valuable information was captured, still some parts were not relevant to the MPC. We concluded that the checklist should be tailored to the developed system and be written in less professional terms to be efficient. In addition an experienced facilitator is very important to the success of this method.

6.2.3 Task scenarios

This method contribution for MPC system was low for the following reasons:

- The few operational scenarios required for the MPC are obvious for Pilots.
- Due to detailed documentation of task analysis, documenting scenario didn't seem to add value.

It was concluded that this technique was not so relevant for MPC. Another Trial will be conducted on an avionics project to evaluate the technique's relevance to LAHAV

6.2.4 Paper prototyping: Task analysis

This method was also new to us. We realised during its planning stage that it needs significant tailoring for our needs and we did that. We wrote down on sticky notes every user function anyone could think of. The sticky notes were logically grouped. After they were grouped the hierarchy was developed. This was done dynamically during the meeting and took several iterations. The functional hierarchy changed significantly and was agreed upon. As a consequence the system architecture has been modified accordingly. The method had a great impact on the MPC software look and feel as well as it's software requirements and architecture.

6.2.5 Evaluate usability of existing system

Four users evaluated the existing system. Each user was given short (15 minutes) training on the system. The user was given a mission to prepare and commented as he went along. Comments were captured by the facilitators generating a detailed list of about fifty problems . The problems were reviewed by the pilots defining the new system to find ways to avoid them in the design of the new system. The users filled out a SUMI satisfaction questionnaires after the evaluation (see 6.3.1 for details).

The technique was very productive though has been applied in a semi-formal way. A more formal trial (more training, better instructions, more users) is being considered.

6.2.6 Set usability requirements

Goals for task time were agreed, and a list of potential user errors were identified. We realise the need for the technique and it's potential but more work is needed to better define it.

6.2.7 Paper prototyping of screens

We haven't used this method before and had doubts about it's value, mainly because it is now very easy to create computerised UI prototypes. It turned out to be a false doubt and the potential users and developers liked the method and its contribution to MPC usability. Mockups of screens were posted on the wall and provided the "Big Picture", although were too small to see the detail. Each screen was displayed using an overhead projector resulting in very fruitful and productive discussions by potential users. A detailed list of 23 usability comments was created.

6.2.8 Style guides

Off the shelf style guides were provided to the developer. It turned out that these style guides are very detailed and difficult to use. Given intuitive visual development tools, developers prefer to learn by click and see rather than reading lengthy manuals.

We realise the need for a style guide, but currently don't have a good one. Good style guide in our view should:

- Be at the appropriate (to the developers) level of detail
- Not to be over restrictive (Leave some space for creativity)

It is still an open issue at LAHAV.

6.2.9 Evaluate Usability of Computer Prototype

The system was only partially developed. But the UI was complete and the main modules were working. General training was held at the beginning for the users resulting in some comments that were captured by the facilitators.

Each user received instructions regarding the mission he had to plan, and worked without assistance. The user spoke freely during the evaluation and the facilitators documented all comments. Software developers were present and observed the evaluation. In general the developers were very receptive and co-operative. Nevertheless towards the end of the evaluation they seemed to lose patience.

A summary meeting was held at the end of the evaluation. Comments were listed and prioritised. It was agreed to fix 93 of the 97 problems. The problems were points of detail and not major issues showing that earlier design was sound.

6.2.10 Test Usability against requirements

Major MPC parts were completed. The system was tested against timing requirements defined for two typical tasks (see 6.2.6).

Eight pilots including fighter pilots, helicopter pilots and navigators participated in this technique.

First, MPC frontal familiarisation training was held for all pilots (2 hours) following by individual hands-on practice for another two hours.

Each pilot received written instructions regarding the mission he had to plan and modify, and worked without assistance. He also could write down comments on collection of printed screens.

The facilitators and developers observed the work on the repeater display and documented their observations. The time was recorded for completion of each task.

Following the completion of both tasks, each pilot completed his comments on printed screens, filled up the SUMI satisfaction questionnaire and explained his comments and impression to the facilitators. All pilots were happy with the MPC as also can be seen from SUMI results.

The tasks performance duration were according to requirements with one exception as explained in section 6.3.2, and some interesting observations could be made.

A summary meeting was held at the end of the evaluation. Comments were listed and prioritised. It was agreed to incorporate 39 of the 54 comments. Seven comments were not accepted and another eight undecided. The problems were points of detail and not major issues showing that earlier design was sound.

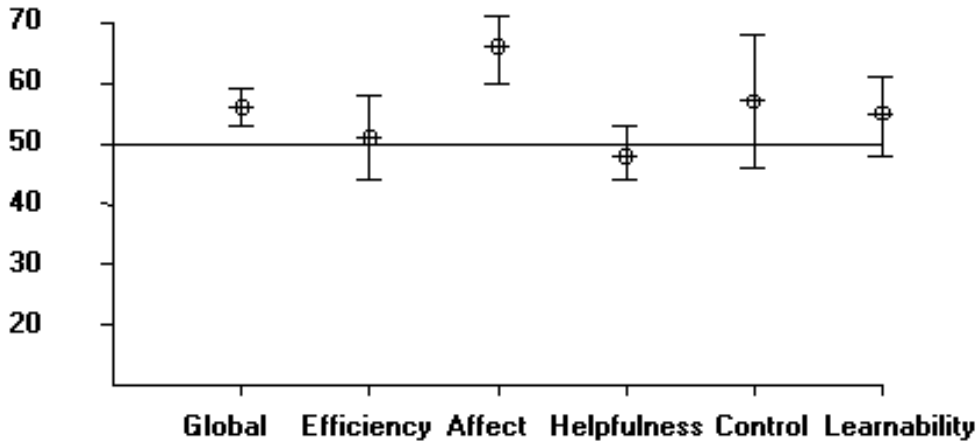
The technique was very productive.

6.3 Improvement in usability

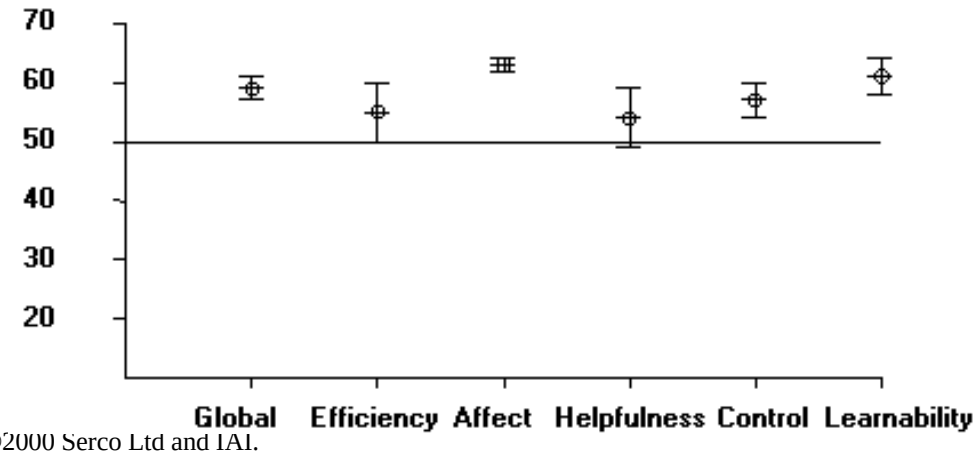
6.3.1 Satisfaction

The charts below show the SUMI results for the evaluation of the first prototype of the new MPC, and the final usability test of the new MPC. The bars show the 95% confidence limits.

Evaluation of new MPC prototype, December 1999



Usability test of new MPC prototype, July 2000



SUMI is scored in relation to an industry average of 50, with scores of ±10 representing one standard deviation. So two thirds of all SUMI scores are in the range 40 to 60.

The usability requirement was for a SUMI score greater than 50. The overall scores for the evaluation and test of prototypes of the new system were well above the industry average at 56 and 59. The profiles of scores for the two evaluations of the new system were similar: it was very strongly liked (affect), and users found it easy to learn and felt in control. They did not find it so helpful (but the help system had not been completed) and they did not feel so efficient (but efficiency might be expected to increase with repeated use of the system). These scores were much better than for the existing MPC.

Overall the above-average SUMI results are very good for a users' first experience of a prototype system.

6.3.2 User performance

The usability requirements established were not more than 40 minutes for the main task and not more than 20 minutes for the secondary task.

All the pilots completed the task within the planned time (except a planner who chose to carry out the task in a more thorough way than a normal pilot would). Navigators were faster because they have more experience of carrying out this task. Nevertheless, the typical time for a pilot to carry out the task is two to three times as long as an expert. This should decrease as pilots become familiar with the system. If not, the possibility of making further usability improvements should be investigated.

6.4 Improvement in usability maturity

The overall ratings for each Usability Maturity Model process are given below, and show a very significant improvement, meeting the objectives set in the first assessment:

Process	**1st assessment**	**2nd assessment**
1 Ensure HCD content in system strategy	**P**artly	**L**argely
2 Plan and manage the HCD process	**P**artly	**L**argely
3 Specify the stakeholder and organisational requirements	**N**ot done	**L**argely
4 Understand and specify the context of use	**L**argely	**F**ully
5 Produce design solutions	**P**artly	**L**argely
6 Evaluate designs against requirements	**N**ot done	**L**argely
7 Introduce and operate the system	**X** Not in the scope of the assessment	**X** Not in the scope of the assessment

6.5 IAI Conclusions

After application of the techniques, the pilots group assessed the benefits. The conclusions were very positive.

- Most of the techniques are very intuitive to understand, to implement and even to facilitate. The techniques are divided into two major categories: (1) meetings or workshops usually lasting 2-6

hours with about 3-6 participants. (2) a one on one paper or computer prototype evaluation by potential users, about 2 hours for each one.

- Practising these techniques in the early stages of design and development ensured less design mistakes later on.
- All participants and developers thought that most of the techniques were worthwhile and that they helped in developing a better and more usable system.
- The techniques were assessed as very cost effective and low cost.

The last observation deserves elaboration. Usually introducing changes into an organisation is a lengthy, costly and complicated process. It requires convincing many people to invest time and money and then demonstrate the benefits versus costs. In the recent years it became even more difficult due to staff shortage and the requirement to reduce the time to market.

TRUMP was the exception due mainly to its low cost, and obvious benefits. When the developers only have to invest a few days in applying the methods and see the results on the spot, convincing the managers is very simple and performing cost-benefit analysis is simply not needed.

In view of the short time and effort it took to practice these techniques and the strong impact they had on the quality of the system, they are being incorporated in LAHAV's development process. The expertise available at LAHAV to practice these techniques is not great. Nevertheless the techniques are fairly intuitive and should be easy for new facilitators to learn.

We are currently working on establishing a specific support structure for disseminating the techniques into other IAI divisions.

7. General conclusions

In many respects the results obtained in the trial at the Inland Revenue were similar to those at IAI. In both organisations the usability maturity model was a valuable tool for identifying needs for process improvement. The Inland Revenue valued the detailed information obtained from a summative assessment requiring three person weeks effort, while for the smaller development group at IAI many of the benefits were gained from a simpler formative one-day assessment.

Particular user centred design methods were not of equal value to both organisations. For example, IAI staff were much more familiar with the usage environment, so that context of use and scenarios were of less benefit than at the Inland Revenue, where they were important in establishing a common understanding. So methods need to be selected and tailored to meet the needs of the development environment.

From the experience gained in the two organisations Serco has developed the general-purpose methodology incorporating described in section 4. These methods implement the principles of ISO 13407, and should be sufficient for many development environments. In some cases they should be complemented by other more specialised methods. More details can be found on the TRUMP web site www.usability.serco.com/trump.

8. References

Bevan N (1999) Quality in use: meeting user needs for quality. In: Journal of Systems and Software,49(1), 89-96.

Bevan, N and Bogomolni, I, Ryan, N (2000) Cost-effective user centered design. Submitted for publication.

Bias, G. and Mayhew, D. *(Eds)* (1994). *Cost-Justifying Usability.* Academic Press.

Earthy, J (1998b) Usability Maturity Model: Processes. INUSE deliverable D5.1.4p, http://www.lboro.ac.uk/eusc/index_r_assurance.html

IBM (1999) Cost Justifying Ease of Use http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/23

ISO 13407 (1998) User centred design process for interactive systems.

ISO/IEC 14598-1 (1998) Information Technology - Evaluation of Software Products - Part 1 General guide.

ISO TR 18529 (2000) Human-centred lifecycle process descriptions

Keil, M, & Carmel, E. (1995). Customer developer links in software-development. Communications of the ACM. 38(5), pp33 - 44.

QWE2000 Session 11I

Mr. Bob Bartlett [UK]
(SIM Group Limited.)

"Experience Testing E-commerce Systems (11I)"

Key Points

- Stress
- Performance
- Security
- Available Tools

Presentation Abstract

The format of the presentation will be a tutorial based on class studies and actual experiences of the author. Results, observations and recommendations will be put forward. The audience will come away with a good understanding of the kind of approaches required to test industrial strength eBusiness solutions.

About the Speaker

Bob is the Chairman of SIM Group Ltd. SIM specializes in Software Testing and has put in place a number of highly efficient testing systems that automatically test sophisticated and mission critical software systems. SIM is the UK leader in Providing efficient solutions for software testing. SIM's work has had a profound impact on the way companies approach testing and improvements to testing have been realized with SIM's help. Bob has over 30 years of software experience using automated testing techniques. He is the Executive Director and Chairman of Software testing specialist company today. He is also a member of the CSSA executive council and has designed, developed and sold automated testing tools. Bob, a manager of major software development and implementation projects, is a test adviser to some of the largest testing projects taking place in U.K. Bob has Trained and lectured in automated testing and software testing techniques,has a track record for substantial reductions in time and cost to test, and successfully managed the growth of start up companies throughout his career.


GRESHAM
Experiences testing
eCommerce systems
Bob Bartlett
bob@simgroup.co.uk
SIM GROUP
© 2000, SIM Group Ltd..


Who we are ….
SIM GROUP
GRESHAM
Who are we?
Independent testing organisation
Specialists in testing for 10 years
Largest implementor of automated testing in UK
Full Service testing Solutions
What do we do?
Testing Projects
Hosted Testing
Testing Consultancy
Implementors of automated testing for over 10 years
Methods, procedures and strategies for efficiency & effectiveness
© 2000, SIM Group Ltd..

Agenda

- *What is the commercial case for quality web sites?*
- *The 12 Commandments for a good web site*
- *Testing requirements*
- *Types of testing*
- *Test processes*
- *Test Automation and Continuous testing*

Quality web sites - WHY bother?

- *First to market is first in the mind.*
- *Having a low percentage of the market with a poor quality web site is better than 0% of the market with NO web site!*
- *Visitors will tolerate imperfections if the business proposition is compelling enough.*
- *There is no time AT ALL to test.*

But on the other hand … Quality does cost!

- *There are numerous examples where quality problems caused misuse of web sites - costing the owners millions of pounds!*
- *More than ever branding and brand images are important - web sites contribute to the definition and qualification of a brand.*
- *Research shows that most people abandon web sites for quality reasons - and they are unlikely to return.*

The 12 Commandments

1. *Speed and performance*
2. *Access and availability*
3. *Up to date and accurate information*
4. *Responsiveness visible and apparent*
5. *Tracking of the business conducted*
6. *Customer service is consistent*
7. *Feedback channels work*
8. *Search and intuitive menus*
9. *Real time processing when beneficial*
10. *Verification of business terms*
11. *Presentation and usability*
12. *Security of site, function and data*

Important Requirements for testing ecommerce systems

- *24 x 365*
- *Loads impossible to predict*
- *Most sites are an integration of many components from many suppliers*
- *Sites get developed and put in place incrementally and iteratively.*
- *All in the public domain*
- *When something goes wrong - how do you know it has gone wrong?*
- *Some testing WILL happen after the site is live*

In the beginning …. Testing was …..

- *Functional*
- *Function and Data security*
- *Load and Stress*

In the early 90's we need more testing ....

- *Functional*
- *Function and Data security*
- *Load and Stress*
- *Integration*

This was mostly caused by the move to Client Server and integrated application solutions.

Not much later we began another type of testing

- *Functional*
- *Function and Data security*
- *Load and Stress*
- *Integration*
- *Presentation*

This was driven by GUI systems

With web testing we added four more types of testing ...

- *Functional*
- *Function and Data security*
- *Load and Stress*
- *Integration*
- *Presentation*
- *Content*
- *Usability*
- *Compatibility*
- *Site Security*

tSpectrum

Alternatively - Looking at a typical project

	Dev.	Link	Sys.	Usab.	Tech	UAT	Oper
Functionality	√	√	√	√		√	
App & Data Security			√	√	√	√	
Load & performance					√		√
Presentation			√	√		√	
Integration		√	√		√		√
Usability				√	√	√	√
Penetration					√		√
Compatibility			√	√	√	√	
Content				√		√	

Stakeholders involved in testing

- ***Ownership of the site***
  - Business responsibility and ultimate responsibility for success of the site
- ***Designer & Developer***
  - Designer - Look & feel
  - Developer - technical and functional design
- ***Integrator***
  - The whole site
  - Back end integration
- ***Independent Testing***
- ***Test Lab Testing***

What should the owner test?

- Functionality → Key business transactions and events
- App & Data Security
- Load & performance
- Presentation
- Integration
- Usability
- Penetration
- Compatibility
- Content → Accurate information

What should the designer / developer test?

SIM GROUP GRESHAM

- Functionality → Does it work?
- App & Data Security → Security features
- Load & performance
- Presentation
- Integration
- Usability
- Penetration
- Compatibility
- Content

What should the integrator test?

SIM GROUP GRESHAM

- Functionality
- App & Data Security
- Load & performance
- Presentation
- Integration → Do all of the pieces fit together and work?
- Usability
- Penetration
- Compatibility
- Content

What should the independent tester test?

SIM GROUP GRESHAM

- Functionality → Does it do what it is suppose to?
- App & Data Security → Is everything secure?
- Load & performance
- Presentation → Does all presentation work?
- Integration → End to end testing.
- Usability
- Penetration
- Compatibility
- Content

What should the test lab test?

SIM GROUP GRESHAM

- Functionality
- App & Data Security
- Load & performance → Special hardware and skills needed
- Presentation
- Integration
- Usability → Target user representatives needed.
- Penetration → Special software and skills needed.
- Compatibility → Wide variety of access needed.
- Content → Continuous testing on content changes

Automated Testing and Continuous Testing

SIM GROUP GRESHAM

- *Automation is more attractive than ever, but even harder to implement with reduced time frames for testing and constantly changing presentations.*
- *Automated load and stress testing is an absolute necessity.*
- *We have found automated functional and automated regression testing is possible using table driven automated testing techniques.*
- *Continuous automated tests are selected from automated test library.*

Methods of testing

SIM GROUP GRESHAM

	Manual	Automated	Continuous
Functionality	√	√	√
App & Data Security	√	√	√
Load & performance		√	√
Presentation	√	√	
Integration	√	√	√
Usability	√		
Penetration	√	√	√
Compatibility	√	√	
Content	√	√	√

To summarise ...

	Owner	D & D	Integrate	Ind.	Test Lab
Functionality	√	√		√	
App & Data Security		√		√	
Load & performance					√
Presentation				√	
Integration			√	√	
Usability					√
Penetration					√
Compatibility					√
Content	√				√

Bob Bartlett
bob@simgroup.co.uk
www.simgroup.co.uk

QWE2000 Session 11M

Mr. Vassilios Sylaidis, Mr. Dimitrios Stasinos, Mr. Theodoros [Greece]
(INTRACOM S.A.)

"Software Development Process Improvement For Telecommunications Applications By Applying Gilb's Inspection Methodology"

Key Points

- Software Process Improvement Experiment
- Defect Detection and Prevention capability enhancement
- Reducing (in a measurable way) reliance on testing for achieving software quality.

Presentation Abstract

This paper describes acquired experiences from the ESSI Process Improvement Experiment GINSENG (Gilb's Inspections for Software Engineering, #27275, 1998-June 2000), funded by the European Union. Its objective was to establish at Intracom's Software Design Centre (SWDC) a systematic framework for software inspections based on Gilb's Inspection Method. Through this inspections framework, Intracom aimed to improve its current practices for telecommunications and other embedded software development by increasing the effectiveness of early defect detection and prevention activities. Additionally, suitable inspections measurements support improvements in the inspection and software development processes and lead to reduced reliance on testing. Results obtained from introducing the method to a typical software development project used as baseline were favourable and justify the expectations and investments and plans are being implemented to internally spread the practice.

Training was obtained by Tom Gilb himself, a world renown expert and consultant in S/W Engineering, Management and Quality issues, as well as in a UK firm which had pioneered the Gilb Inspection Method, where the field experience was also evaluated.

The initial step of GINSENG consisted of introducing the experiment within a digital telephone switch baseline project implementing new functionality in an incremental way. Thus, training in Gilb's Inspection Method was carried out and the inspection procedures for the baseline project were documented, by adapting Gilb's method to the standard development procedures used which ran in parallel with the baseline projects' implementation of early increments. Inspections were then executed throughout the baseline project's latter stages (increments). PIE results were

evaluated based on appropriate measurements. Finally, the appropriateness of Gilb's inspection method for Intracom's software development environment (mostly developing embedded systems) was evaluated, leading to plans for further internal exploitation which has been started.

Improvements to date mostly relate to higher emphasis and efficiency of the defect detection process, while defect prevention (i.e. avoidance of injecting defects in the first place, i.e. in software design) will be facilitated in the future based on specific activities and infrastructure built.

Gilb's Inspection broad interest and wide applicability support the transferability of the GINSENG experiences. Internal dissemination actions address Intracom Group of companies, while external ones Greek SMEs and European and international firms, since the business problems addressed were not unique to INTRACOM but applies to a very wide audience concerned with Software Development.

About the Speaker

Mr. Vassilios Sylaidis before joining INTRACOM S.A. in 1991, was employed in the Hellenic Navy Research Centre as a system and software engineer. Besides the program described here for which he provided expert support and performed a monitoring role, he also previously coordinated adapting the ISO 9001 certified Quality System for Software development as well as he was champion in introducing various Process Improvements, s.a.: the CMM framework, AMI metrics management, PSP, also planning P-CMM etc. Also participated in international programmes in software quality area, as well as evaluated and reviewed EU funded proposals and programs in the software IT area. He was also INTRACOM's project manager for successfully implementing a Y2K compliance programme (for contact data see above).

Mr. Dimitris Stasinos, MSc, an expert telecommunications S/W development engineer in INTRACOM, was trained as an inspection expert by Tom Gilb and acted as a facilitator for the introduction of Gilb's inspections in INTRACOM. He is also currently responsible for implementing improvements by applying the Software CMM improvement framework for which he was specially trained.

Mr. Theodoros Karvounidis, is currently a project manager for a telecommunications software development project. He is also an experienced telecommunications software designer. He was project leader responsible for the introduction of Gilb Inspection Methodology in Intracom's Software Design Centre. He has been trained at the SEI in the US as a trainer in Watt Humphrey's Personal Software Process and has delivered courses for INTRACOM employees on the subject.

QWE2000 Vendor Technical Presentation VT 9

Mr. Peter Sterck [Belgium] (ps_testware)

The Challenge of e-business Testing

Key Points

Key Points to be supplied.

Presentation Abstract

If you want to make money in business, you have to comply with a set of rules. Three rules require special attention within the context of software testing an e-business solution. Present the benefit of the offering to the customer. Under all circumstances, this presentation must be KISS (keep it simple and stupid), in e-business it must be e-KISS, extremely - KISS. The challenge is usability.

The need for a smooth purchasing process. If you are confronted with a closed shop door or have to queue up longer than expected, you are inclined to look for alternatives. In e-business, the alternative is only one click away. The challenge is availability and performance of the site. Confidence in the transaction and the people involved. You do not give your money to anybody, you do not even accept money from everybody. With e-business you do not have a voice, a face or a reference to check. The challenge is security.

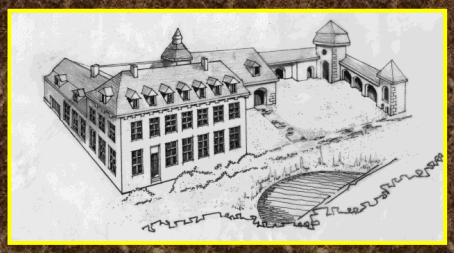
I know that you are already aware of these challenges, but what are you going to do to avoid making a (big) mistake? Having know-how and experience in each of the domains is too much luxury for most companies. Come and talk to us.

About the Speaker

Peter Sterck started his career at CMB, a worldwide shipping company, where he was responsible for the software that managed the container logistics. In 1992 and 1993, he acknowledged the extreme importance of software quality and joined Performance Software where he was a sales account manager. With the support of Performance Software he established his own company in1993. At first, the company concentrated on selling testing tools but soon experienced that tools should be used with a proper testing method. The method that was developed within the company was based on the V-model of Glenford Meyers. Since then, Peter Sterck has been responsible for the growth of ps_testware from 9 till 35 co-workers

within 3 years. He is an acknowledged speaker when it comes to software testing. Together with companies such as Oracle, HP, Sybase, Mercury Interactive and Rational Software he increased the awareness of the need of Structured Software Testing during several seminars and conferences. Peter Sterck received his degree of Electronic Engineer in Leuven, Belgium, and his MBA in Louvain-la-Neuve, Belgium.

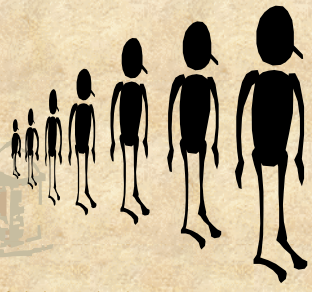
ps_testware



The Challenge of E-business Testing

The Challenge of E-Business Testing

Peter Sterck
CEO

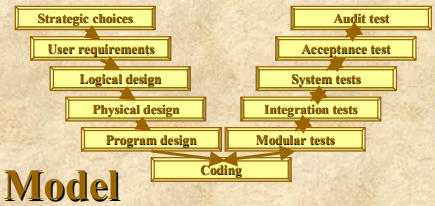


The Challenge of E-Business Testing

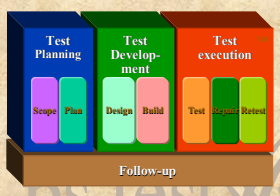
- Challenge 1 : Usability
- Challenge 2 : Availability and Performance
- Challenge 3 : Security

Testing Software : The Reference

- Structured Software Testing
- Methodology



- Implementation Model



Challenge 1 : Usability

Is the e-business site easy to use?

- Ease of learning
- Efficiency of use
- Memorability
- Error frequency and severity
- Subjective satisfaction
- Correct functionality

Chal 2 : Availability and Performance

Is the e-business site accessible and fast?

- Portability
- Reliability
- Scalability
- Download time
- Breakdown Test
- Monitoring
- Tools
- 24x7
- Real World

Challenge 3 : Security

Is the e-business site safe for seller and buyer?

- Continuity, Integrity, Confidentiality
- Never ending effort
- Break-ins, Sniffing, Denial-of-service attack
- Trojan Horse, Virus
- Server, Network, connection, client
- Firewalls, Encryption, Authentication
- Security scanners and auditing tools

ps_testware
Software Testing Services

Tiensesteenweg 329
B-3010 Leuven
Tel.: +32 (16) 35.93.80
Fax: +32 (16) 35.93.88
e-mail: info@pstestware.com
http://www.pstestware.com

ps_testware
Software Testing Services

QWE2000 Session 12T

Mr. Sanjay DasGupta & Indrajit Sanyal [INDIA]
(Usha Communications Technology)

"A Java-XML Integration for Automated Testing (12T)"

Key Points

- XML
- Java
- Scripting

Presentation Abstract

The proposed paper describes an unusual application of XML: to represent and run test scripts for software written in Java. There are two aspects to this: an XML convention (enshrined in a DTD) for the representation of program scripts, and a Java executable (archive) that can interpret them. The archive can be used with (or to build) a test harness that performs tests automatically. The scripts and matching run-time functionality are capable of representing ôhowö something is done with flexibility comparable to any 3 rd generation language (specifically Java). The proposed paper describes capabilities and design of the Jeeves system and highlights its applicability in the automated testing arena. It includes a short tutorial and describes scenarios to illustrate the advantages of using it.

About the Speaker

Mr. DasGupta is a consultant in the Engineering department of Usha Communications Technology, playing the role of Chief Architect for the product development, and is located in companyÆs R&D facility in Calcutta, India. With more than 15 years of experience in software development prior to joining Usha Comm, he was Product Director for Enterprise Management products with a Computer AssociatesÆ subsidiary in India. Mr. DasGupta also held different strategic positions in several international software companies in India and Europe. Mr. DasGupta holds a BS degree in Chemical Engineering and a MS degree in Software Systems.

Indrajit Sanyal is a Group Manager in Engineering department of Usha Communications Technology and is located in companyÆs office in Calcutta, India. In this role Mr. Sanyal overlooks internet and web oriented strategic development projects. He has been involved with software design and development for telecommunications, simulation and graphics, and industrial automation for over 8 years. He earned his BS degree in Mechanical Engineering from the Indian Institute

of Technology at Kharagpur, India.

A Java-XML Integration for Automated Testing

Sanjay DasGupta & Indrajit Sanyal

Ushacomm India Ltd.
www.ushacomm.com

Topics

- Programming Languages in Testing
- Embeddable Scripting Languages
- Java and XML-Based Scripting
- The Design of Axess
- Using Axess
- Applications of Axess
- Conclusions

A Java-XML Integration for Automated Testing

CONTRIBUTORS:
Sanjay DasGupta, sanjay.dasgupta@ushacomm.co.in
Indrajit Sanyal, indrajit.sanyal@ushacomm.co.in

1. Introduction

This paper describes a Java [1] tool that can interpret and *run* programming scripts written in a language based on the *Extensible Markup Language* (XML) [2]. The software and XML extension are referred to as *Axess* (An extensible, embeddable scripting system for Java). The Axess software can load and use programming scripts at run-time, thus allowing scripts to be embedded in other programs. This aspect allows Axess' capabilities to be easily leveraged for other applications, especially test automation.

The advantages of using XML for representing a programming script include the ability to use a wide variety of tools and approaches for creation, examination and modification. This tools-based approach and the choice of XML as the basis for a scripting language provides many possibilities. This paper discusses the design of Axess and shows how its features support the objectives of tools-based scripting and seamless integration with the Java run-time system. We use two case studies to illustrate how an approach based on Axess can help in the creation of automated tests for different domains.

2. Embeddable Scripting Languages

The use of scripting languages in automated testing is well established. There are numerous proprietary systems in use by product vendors, but open-source systems and languages are also used extensively. Case studies on the use of an open language called *Tcl* (the *Tool Command Language* designed by John Ousterhout and now available from Ajuba) can be found at the *Ajuba*[1] website [5]. To be truly effective and have synergy with an application, a scripting language must be embedded with the host application's implementation language. The appeal of Tcl is that it is designed to be embedded in C and can, therefore, be used to seamlessly extend applications (written in C and C++) with a scripting interface. Several open-source scripting languages are now available for use with Java [6]. Some of these languages, such as *DynamicJava* and *BeanShell* [7] are also embeddable.

2.1 Embedding in C

However, the capabilities gained through embedding depend on the design of the language as well as the possibilities offered by the base language and platform. In the case of Tcl, an extremely flexible model makes it possible to integrate the application's C code very tightly with the scripting infrastructure. Application-specific Tcl extensions typically use the following capabilities:

- Map values of script and application (C/C++) variables (so changing the value on one side appears to set the value on the other side as well)
- Set traces on script variables (so a C function is called whenever a script variable is read or written)
- Define new script commands, which cause specified C functions to be called when the command is invoked.

[1] Ajuba recently changed their name (from *Scriptics*), but their website is still at *www.scriptics.com*.

The ability to link script *commands* to specific functions within the application is especially effective and makes Tcl applications extremely powerful.

2.2 Embedding in Java

In the case of Java, most scripting languages leverage capabilities provided by the *java.lang.reflect* package. The reflection API allows Java code to access members of a class (or instance) without using the *dot* notation. It provides utility classes called *Field* and *Method* that can be used to manipulate data and function members respectively. Instances of these classes can be obtained by specifying the name (and signature in the case of functions) of the member. A data member (of a class or instance) can be accessed (read or modify) by calling appropriate member functions on the Field object that represents it. Similarly, a Method object can be used to invoke a function, passing suitable arguments, and receive the return value, if any. The reflection API also provides mechanisms for the instantiation of new objects and the construction of new arrays without using the *new* operator.

What this means to Java scripting languages is that they can use any accessible Java class. These classes do not need any special code to allow them to work with the scripting system.

3. Java and XML

The synergy resulting from the use of XML with Java in an application has been widely discussed and documented [3, 4]. Both Java and XML are platform-independent, open, standard[2] and widely available. Standards for the use of XML documents in Java programs – the *Simple API for XML* (SAX), and *Document Object Model* (DOM) – are stable and universally accepted, and implementations of these APIs are also available from Sun Microsystems, IBM and others. The wide availability and applicability of this combination has spurred activity in many areas and as a result, a variety of open de-facto standards and applications are becoming available.

Despite all the activity concerning Java and XML, there has been no attempt to use XML as a scripting language for Java. The *XSLT* and *XMLScript* efforts involve the use of XML and scripts, but are very definitely targeted for the use of scripts to transform XML into other forms better suited to display devices.

4. Design and use of Axess

In designing Axess, we defined several objectives. The following subsections describe these objectives and the extent the objective is met.

4.1 Compatibility with Java

Axess is written in Java and so can be used with other Java applications. The use of the reflection API ensures that Axess is compatible with the Java run-time system and able to access and use all standard as well as user-installed APIs and classes, just like code in any other compiled Java class.

[2] Java is a registered trademark of Sun Microsystems. Though not yet formally standardized, a large part of the industry accepts Sun's stewardship of Java, and it is possible to obtain products compliant to Sun's blueprints from many sources.

4.2 Java-Like Usage

Axess scripts are not expected to visually resemble Java code (see section 4.4 below). Although in terms of the creation and use of data items, Axess scripts can be used to do everything[3] that a Java program can perform. They can instantiate objects as well as primitive data entities of all available types and perform all the operations available to Java programs, using the same programming constructs.

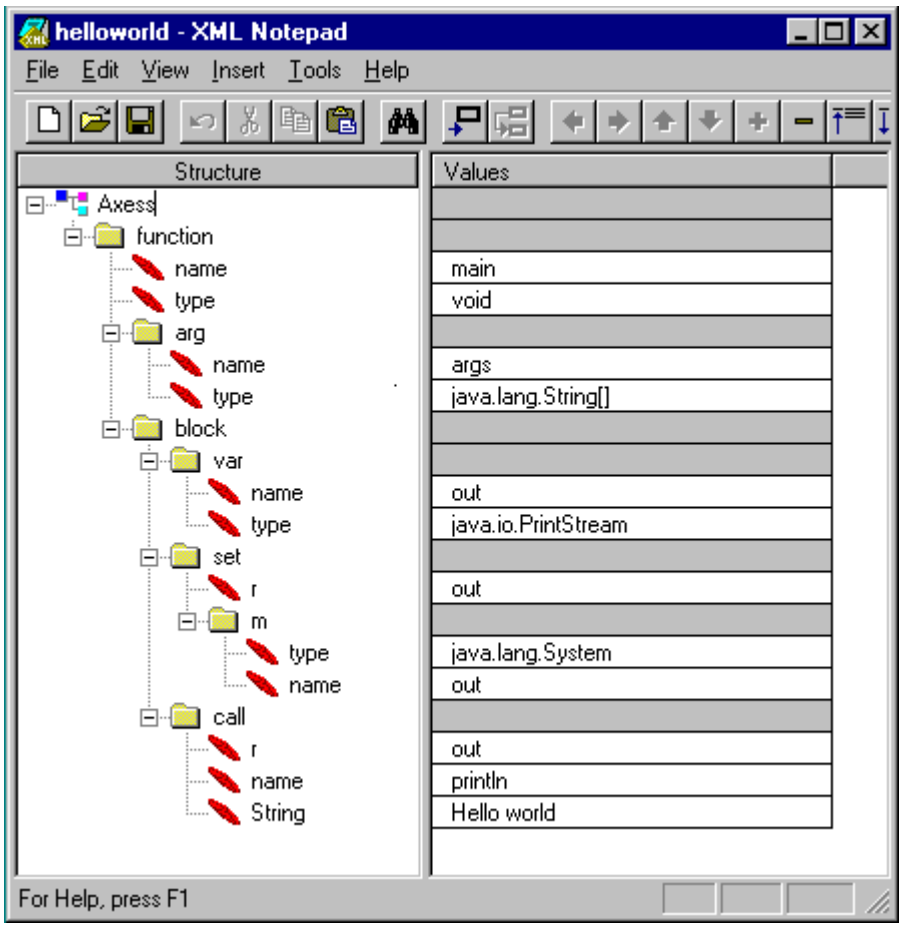


Figure 1. Using Microsoft's *XML Notepad* to display an Axess script

4.3 Efficiency and Compactness

A Java jar file containing the Axess software is less than 80 Kbytes (not including the XML parser, but minimal parsers fit in as little as 10 Kbytes). This makes it suitable to use even as an applet downloaded into a web-browser.

[3] With the exception of defining a new Java class. Other Java-based scripting languages include features that appear to allow the creation of new classes. However, these are merely artifacts with a superficial similarity with real Java classes.

4.4 XML Based

Axess scripts need to be represented in XML to enable a tools-based approach to the creation, analysis and modification of scripts. Figure 1 above shows how the Microsoft *XML Notepad* displays an Axess *Hello world* application.

4.5 Simplicity

Axess is simple and intuitive to use. The tags that are used are as close to corresponding Java keywords as possible. XML has numerous esoteric features such as DTD, Schema, Namespace, etc. However, Axess uses none of them.

5. Experience With The Use of *Axess*

We used Axess in several testing situations, using different applications and handling different aspects of the work. The following subsections describe our experiences using Axess for two diverse applications and evaluate the value of a tool like Axess.

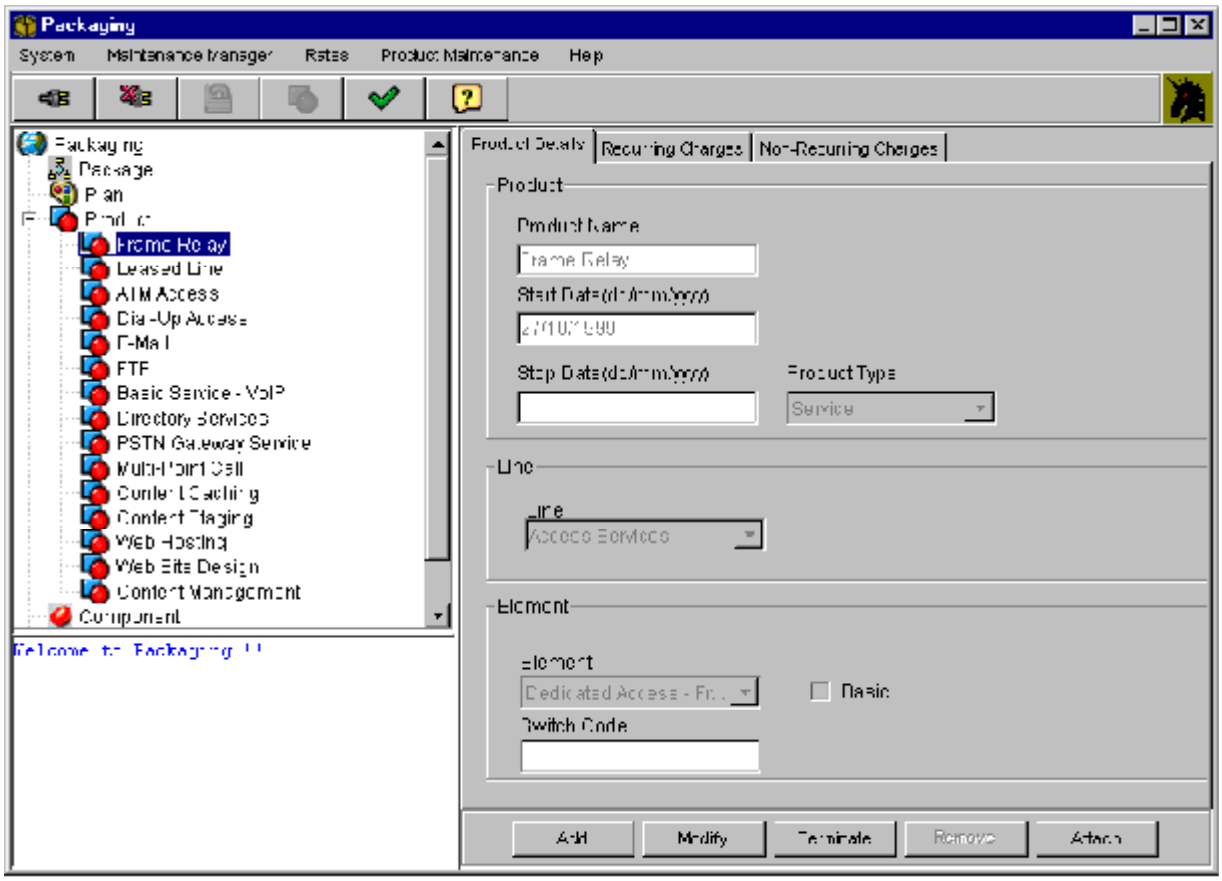


Figure 2. A typical application GUI

5.1 Toolkit for GUI-Driven Tests

Record and replay of GUI inputs is a very common technique used in regression testing. The typical steps used are as follows:

1. Record user interaction and store for later replay.

2. Play the stored information to recreate the test activity.

However, it is also useful to be able to write test scenarios from scratch (rather than to record them) or to be able to edit and modify a recorded scenario. The ability to write the scenario as a script is an advantage, since it can contain logic that is sensitive to the state of the application or the test's environment.

Axess was used to create a custom test harness for a customer care application of a convergent billing system and exercise the following actions:

1. Find a particular customer's record
2. Retrieve customer's service details
3. Attach a specific product to the customer's subscription

Figure 2 above illustrates the complexity of a typical application GUI.

5.1.1 Approach using Axess

The approach used for the test described above required close integration with the Java run-time system, particularly the part used to process GUI events, and required the development of two sets of Java classes – one for recording the GUI actions and another for their subsequent replay.

To record GUI events, an instance of a specialized sub-class of *java.awt.EventQueue* is created and pushed onto the system event queue:

```
Toolkit.getDefaultToolkit().getSystemEventQueue().push(axessEventQueue);
```

The new class' *getNextEvent()* function is overridden to add the new functionality required – the *awt* event normally returned by it is also processed and stored as an element of a test script in the form of function calls to a GUI robot.

For replaying scripts, the *java.awt.Robot* class is used. Unfortunately, the *Robot* class is only available in recent versions (1.3) of Java, thus precluding its use with applications developed using earlier versions of Java.

5.1.2 Generated Script

The text in Figure 3 below is an extract of a script file generated by the GUI testing toolkit.

```
<call> <name>auto.RoboUser.init</name> </call>
    . . .
<call> <name>auto.RoboUser.clickOnXY</name> <i>286</i> <i>219</i> </call>
<call> <name>auto.RoboUser.delay</name> <i>1406</i> </call>
<call> <name>auto.RoboUser.typeInText</name> <s>151</s> </call>
<call> <name>auto.RoboUser.clickOnXY</name> <i>416</i> <i>223</i> </call>
<call> <name>auto.RoboUser.delay</name> <i>3406</i> </call>
<call> <name>auto.RoboUser.clickOnXY</name> <i>360</i> <i>59</i> </call>
<call> <name>auto.RoboUser.delay</name> <i>7141</i> </call>
<call> <name>auto.RoboUser.clickOnXY</name> <i>272</i> <i>217</i> </call>
```

Figure 3. Form of script generated by testing tool

Observe that the script does not use knowledge of the GUI objects to which the events are directed. In certain situations, this can be a severe disadvantage and while working with an object-oriented language like Java, it must be possible to track object identities and deliver events to specific GUI widgets. In fact, the object-orientation features of Axess (based on Java's object model) allow this to be done and later versions of our tools will leverage these capabilities to provide enhanced capabilities.

5.2 Testing Message-Oriented Systems

Another application automated with Axess is the emulation of a device that provides *voice over IP* (VoIP) usage data to a telecom billing system. Communication with a VoIP device occurs over IP connections using well-defined messages organized in predefined scenarios. The basic approach to such testing uses a Java program (see Figure 4 below) that allows the tester to select a message, populate its fields, and send it to the system under test. The reply message was examined to determine if the response was correct. Scenarios consisting of more that one pair of messages were difficult to setup because the human intervention needed takes time, and can cause one of the protocol state-machines to timeout.

5.2.1 Approach using Axess

The approach was modified using Axess to allow the tester to define configuration data and functions that would allow complete message scenarios to be defined. The number of messages in a complete scenario is not an issue since the tool populates messages automatically using test configuration data and scripts. The tool is also able to check received messages (both type and content), and respond with appropriate messages.

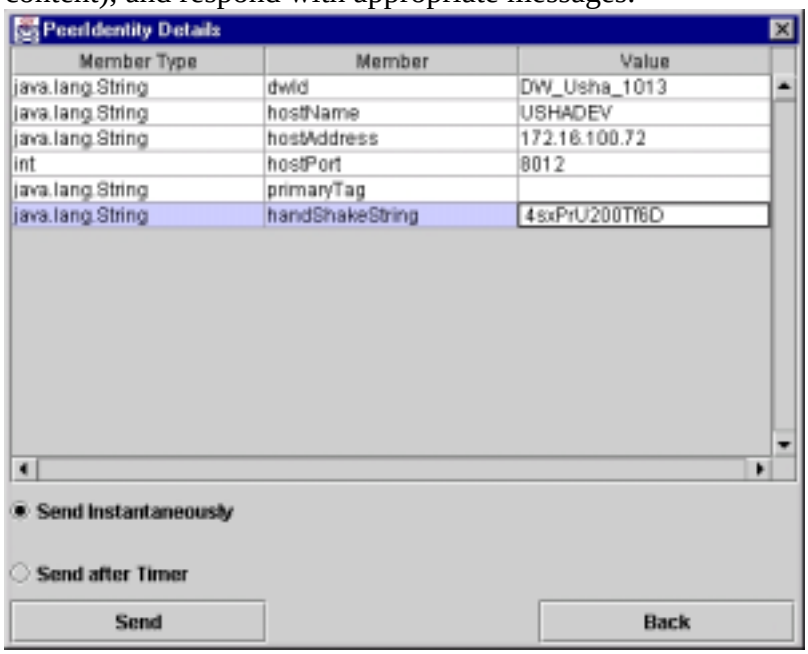


Figure 4. GUI used for selecting and populating messages

6. Conclusions

a) XML is an optimal method to represent and store structured information; it must, therefore, be applicable to programs as well.
b) An XML extension has been designed to represent any Java program. This programming language has been used to represent test scripts used in automated testing.
c) A disadvantage of programming languages based in XML is verbosity. This is particularly apparent when representing Java. However, the use of XML enables a tools-based approach and it will be possible to use formal techniques to prove properties of testing scripts.
d) A compact and efficient Java-based interpreter for XML programming scripts has been created and is used to assist in automated testing of software written in Java.
e) The ability of the interpreter to integrate with test environments and software written in Java is a definite advantage and can be used to raise the level of test automation.

7. Acknowledgements

The authors wish to acknowledge the guidance and encouragement received from Grisha Alpernas, EVP Technical Operations at Usha Communication Technology Inc., and the application ideas of Anirban Mukhopadhyay, Consultant, at Ushacomm India Ltd. This article would not have been a reality without their contributions.

References

[1] *The Java Programming Language, Third Edition*, Ken Arnold, James Gosling and David Holmes, Addison Wesley, June 2000.
[2] *Recommendation "REC-xml-19980210"*, The World Wide Web Consortium, 1998.
[3] *White Paper "Portable Data / Portable Code: XML and Java Technologies"*, (http://www.javasoft.com/xml/nfocus.html), J.P. Morgenthal, Sun Microsystems.
[4] "*XML and Java: The Why and the How*", Israel Hilerio, in Java Developer's Journal, Volume 4 Issue 9, September 1999.
[5] *http://www.scriptics.com/customers/success/testautomation.html*
[6] "*Programming Languages for the JVM*", Rick Hightower, in Java Developer's Journal, Volume 5 Issue 2, February 2000.
[7] "*BeanShell and DynamicJava: Java Scripting with Java*", Rick Hightower, in Java Developer's Journal, Volume 5 Issue 7, July 2000.

Addendum 1: The Axess User Guide

1. Introduction

The Axess software is packaged as a Java *jar* file that can be used to run standalone or as well as embedded scripts (see example in section 2 below) as described in the following sub-sections.

1.1 Standalone Scripts

In the standalone mode, the *Axess* interpreter is used as a Java application, executing script files in much the same waya similar manner that a *shell* script is interpreted. To execute a file containing scripts, it is passed as a command line argument to the *Axess* application:

```
C:> java Axess scriptOne.xml
```

Here As shown above, *scriptOne.xml* is the name of a file containing the script. One function in every standalone script file has a special signature (name = *main*, type = *void*, and takes a single argument of type *java.lang.String[]*), and execution begins hereat that point. The other functions can be called directly or indirectly from *main*. Any command-line arguments (typed after the name of the file containing the script) are passed as arguments to the *main* function.

1.2 Embedded Scripts

Embedded scripts are used in conjunction with Java code in end-user applications. In this mode, Java methods can call script functions, which are passing and receiving data. The script functions can access the Java run-time, and all accessible java classes installed in the system. Functions in the script can also access public members – methods and data – of the invoking object (via a reference passed to it). Embedded scripts are a useful mechanism for enhancing the flexibility and customizability of Java applications. They are also useful for adding code for to traceing and de-bugging applications when needed.

The *Axess* interpreter has a constructor that allows Java modules to create an embeddable module of script functions from a file. Once such an objectthis module has been created, any function in the script can be called by using the method *callFunction*. The following Java code can be used to call the function *scaleDistance* in the script in the example above:

```
Axess module1 = new Axess( new java.io.File( "scriptOne.xml" ) );
float oldDistance, newDistance;
   …
Float oldDst = new Float( oldDistance ), newDst;
newDst = module1.callFunction( "scaleDistance", new Object[] {oldDst} );
newDistance = newDst.floatValue();
   …
```

The method *callFunction* takes two arguments – the name of the script function to call, and a *java.lang.Object* array containing the script function's arguments. The number and type of arguments is used to determine the which script function to call. If the function has arguments of primitive types (float, boolean, etc.), objects of corresponding wrapper types (Float, Boolean, etc.) must be used instead. Any function in the script used to construct *script1* can be called through it.

The embedding Java program can use any number of such *Axess* objects encapsulating different scripts.

The calling thread itself executes the called script function. The method *callFunction* returns an Object containing the value returned by the script function. If the script returns a primitive type, an instance of the corresponding wrapper type is returned.

2. Program Structure

An *Axess* script must have a well-defined structure. The following sample illustrates the organization of a script file:

```
<Axess>

   <function> <name>scaleDistance</name> <type>float</type>
      <arg> <name>oldDistance</name> <type>float</type> </arg>
      <block>
        <var> <name>newDistance</name> <type>float</type> </var>
            …
        <!-- body of function -->
            …
        <return> <r>newDistance</r> </return>
      </block>
   </function>

   <function> <name>…</name> <type>…</type>
       <!-- arguments (in any) and body of function -->
   </function>
      …
      …
   <function> <name>main</name> <type>void</type>
      <arg> <name>args</name> <type>java.lang.String[]</type> </arg>
      <block>
            …
         <!-- body of function -->
            …
      </block>
   </function>

</Axess>
```

The main constructs are *module* and *function* elements. A script file contains a single *module* as the outermost element, which contains one or more functions. All *code* (in a scripting sense) is contained within *block* elements within *function*s as described in the sub-sections below.

2.1 arg (Function Arguments)

The *arg* element is used to define a function argument, and must contain elements to define the name and type of the formal argument as shown in the example below. The name and type must appear in the following order shown:

```
<function> <name>celciusToFahrenheit</name> <type>float</type>
  <arg> <name>celcius</name> <type>float</type> </arg>
  <block>
    …
```

A function may have any number (including 0) of formal arguments., eEach of these is represented by an *arg* element, and placed after the *type* element that defines the function's return type.

2.2 *Axess* (Module Definition)

A *module* element must be used as the outermost containing element of any *Axess* script. Operating system *files* or Java *String*s may contain *module*s. Each module may contain one or more *function* elements.

2.3 *block* Definition

A block is the equivalent of Java's curly braces – *{* and *}*. All of a function's code must be contained within a block.

A block is also used in another context –which is to encapsulate more than one statement into an atomic unit that is executed either entirely or not at all. The following *if* statement illustrates a typical use of a block.

```
<if>
  <lt> <r>count</r> <i>15</i> </lt>
  <block>
    <call> <m> <type>java.lang.System</type> <name>println</name> </m>
        <r>count</r> </call>
    <inc> <r>count</r> </inc>
  </block>
</if>
```

In this example, the two statements (*call* and *inc*) are both executed if the value of count is less than 15, (and neither is executed otherwise).

An empty block, written as *<block/>*, may be used in contexts that require an empty statement (e.g., in a for statement – see example in section 5.7 below).

2.4 *function* Definition

The *function* element is used to define a script function within a module. Each function element contains the following elements:

- *name* (mandatory – 1 occurrence) – specifies the name of the function being defined. The element must contain the name of the function as a piece of text.
- *type* (mandatory – 1 occurrence) – specifies the type of the return value. The element must contain the name of a type as text.
- *arg* (0 or more occurrences) – specifies the name and type of any arguments accepted by the function. The element must contain a *name* and a *type* element each containing text specifying the name and type of the argument respectively.
- *block* (mandatory – 1 occurrence) – contains the variable definitions and executable statements in the body of the function.

A module may contain any number of functions.

3. Variables and Values

Facilities exist for the definition of *auto* variables, which are (defined and used only within functions). All variables are typed; and all of the Java types (primitives, standard JDK classes, user-defined classes and 3rd party classes) may be used.

3.1 Variable Definition

A variable is defined by the *var* element, which must contain two mandatory elements – *name* and *type*. The *value* element is optional, and may be used to specify the initial value for primitive types. The value of an un-initialized variable follows the usual Java conventions. An example of the definition of variables within functions followsis shown below.

```
<var> <name>maxCount</name> <type>int</type> <value>15</value> </var>
<var> <name>appTitle</name> <type>java.lang.String</type>
    <value>Green Grapes</value> </var>
<var> <name>myFrame</name> <type>java.awt.Frame</type> </var>
```

To declare an n-dimensional array, n pairs of box-brackets (*[]*) are appended to the type-name, as shown in the examples below:

```
<var> <name>args</name> <type>java.lang.String[]</type> </var>
<var> <name>chessBoard</name> <type>byte[][]</type> </var>
```

Memory allocation for the elements, however, has to be done separately as explained in section 4.7 below.

3.2 Accessing Primitive Variables

The *r* (reference) element is used to refer to the value of a primitive variable:

```
<var> <name>count</name> <type>int</type> <value>1</value> </var>
    …
<while> <lt> <r>count</r> <int>15</int> </lt>
  <block>
    <call> <name>fillSilo</name> <r>count</r> </call>
    <set> <r>count</r> <add> <r>count</r> <int>1</int> </add> </set>
  </block>
</while>
```

The *r* element can be used to fetch obtain the value of a variable, as well as a *lvalue* to which a new value may be assigned as illustrated by the *set* element in the example above.

3.3 Accessing Members of Java *Objects*

The *m* element (mnemonic for *member*) is used to access a field of a Java object. It must contain two elements – the first a reference to an object whose field is to be accessed, and the second a *name* element that specifies the field to access:

```
<var> <name>xVal</name> <type>int</type> </var>
<var> <name>center</name> <type>java.awt.Point</type> </var>
    …
<set> <r>xVal</r> <m> <r>center</r> <name>x</name> </m> </set>
<set> <m> <r>center</r> <name>x</name> </m>
    <add> <r>xVal</r> <int>128</int> </add> </set>
```

The *m* element can be used to fetch obtain a value, as well as in the form of an *lvalue,* to assign a new value to an object's data-member as illustrated by the *set* in the example above.

3.4 Accessing Members of Java *Class*es

The *m* element (mnemonic for *member*) can also be used to access a field of a Java class (and in this case, the specified data-member must be *static*). In this useinstance, the *m* element must contain two elements – the first a *type* element specifying the class whose field is to be accessed, and the second a *name* element that specifies the field to access:

```
<var> <name>out</name> <type>java.io.PrintStream</type> </var>
    …
<set> <r>out</r> <m> <type>java.lang.System</type> <name>out</name> </m>
    </set>
    …
<call> <r>out</r> <name>println</name> <String>Hello world</String>
    </call>
```

The *m* element can be used to fetch acquire a value, as well asand in the form of an *lvalue,* to assign a new value to a class' data-member.

3.5 Accessing Elements of Java Arrays

The *x* element (mnemonic for *index*) is used to access an element of a Java array. As its first argument, Tthe *x* element must contain a reference to an array (whose element is to be accessed.) as its first argument. References to integers, (eachwhich representing an index,) follow the array reference. The number of index values must not exceed the number of dimensions:

```
<var> <name>temp</name> <type>float</type> </var>
<var> <name>studentAges</name> <type>float[]</type> </var>
<var> <name>i</name> <type>int</type> </var>
    …
<for>
  <set> <r>i</r> <int>0</int> </set>
  <lt> <r>i</r> <int>15</int> </lt>
  <inc> <r>i</r> </inc>
  <block>
    <set> <r>temp</r> <x> <r>studentAges</r> <r>i</r> </x> </set>
    <set> <x> <r>studentAges</r> <r>i</r> </x>
        <mul> <r>temp</r> <float>12.0</float> </mul> </set>
  </block>
</for>
```

The x element can also be used as an *lvalue* (first argument of a *set* element) as shown by in the last example above.

3.6 Garbage Collection

All variables defined within script functions are associated with a dynamically created instance of any appropriately typed object. References to the dynamically created objects are erased when the block within whichcontaining a defined variable is defined is exited.

3.7 Literals

Literal constants of every Java primitive type may be defined using the elements in the table below. The value of the constant is represented by the contained text without noany extra surrounding blanks.

Type	Example
boolean	`<boolean>true</boolean>, <boolean>false</boolean>`
byte	`<byte>128</byte>`
char	`<char>Y</char>`
double	`<double>3.14159</double>`
float	`<float>0.25</float>`
int	`<int>25</int>`
long	`<long>100000</long>`
short	`<short>255</short>`

4. Operators

Based on the Java model, Sseveral operators are implemented., based on the Java model. The following sub-sections detail the behavior of these operators.

4.1 Assignment

The *set* operator can be used to assign a value to *Axess* variables, static or non-static data-members of Java objects or classes, and elements of Java arrays. It takes two operands – the first is an *lvalue* whose value is to be set, and the second an expression that can be evaluated to a value.

4.2 Cast

The *cast* operator can be used to pass a value of one type into a context that requires a value of another compatible type. The cast operator is most useful for argument passing, as in the example below which illustrates (where the run-time function overload resolution would fail to find the function *put* unless the *cast*s were used):

```
<var> <name>hash</name> <type>java.util.Hashtable</type> </var>
    …
<call> <r>hash</r> <name>put</name>
    <cast> <type>java.lang.Object</type> <String>LANGUAGE<String> </cast>
    <cast> <type>java.lang.Object</type> <String>Java<String> </cast>
</call>
```

4.3 Arithmetic

As in Java, Aarithmetic operators as in Java are provided for addition, multiplication, subtraction, division and modulo extraction. The operations perform type conversion as in Java. The following examples illustrate the use of XML elements to represent arithmetic operations.

Java	XML	Example
*	mul	`<mul> <r>celcius</r> <float>1.8</float> </mul>`
+	add	`<add> <r>prod</r> <float>32.0</float> </add>`
/	div	`<div> <r>sum</r> <r>count</r> </div>`

-	sub	`<sub> <r>price</r> <r>disc</r> </sub>`
%	mod	`<mod> <r>position</r> <int>8</int> </mod>`

4.4 Bit Shift

As in Java, Bbit shift operators as in Java are provided. These operators take two arguments, and return a value obtained by shifting the first operand by the number of bits specified by the second.

Java	XML	Example
<<	shiftl	`<shiftl> <int>0x01</int> <r>offset</r> </shiftl>`
>>	shiftr	`<shiftr> <r>status</r> <int>3</int> </shiftr>`
>>>	shiftur	`<shiftur> <r>pattern</r> <r>count</r> </shiftur>`

4.5 Comparison

As in Java, Ooperators as in Java are provided for comparing values of numeric and *char* types (except *eq* and *ne*, which can be used to compare objects as well). The operations perform type conversion as in Java. The following examples illustrate the use of XML elements to represent arithmetic operations.

Java	XML	Example
==	eq	`<eq> <r>diameter</r> <float>2.0</float> </eq>`
!=	ne	`<ne> <r>ptr</r> <r>firstElement</r> </ne>`
<	lt	`<lt> <r>x</r> <int>25</int> </lt>`
<=	le	`<le> <r>error</r> <float>0.0001</float> </le>`
>	gt	`<gt> <r>rate</r> <float>25.5</float> </gt>`
>=	ge	`<ge> <r>personsAge</r> <int>75</int> </ge>`

As in Java, *eq* and *ne* check for object identity – and so should not, for example, be used for comparing two strings. The *equals()* function should be used for these situations. The comparison operators are typically used in conjunction with *if*, *while* and *for* statements.

4.6 Operator instanceof

This operator is used to check if an entity is of a particular type. It takes two arguments – a reference supplying the entity to be tested, and a *type* element naming the type to be tested against:

```
    …
<arg> <name>obj</name> <type>java.lang.Object</type> </arg>
<if> <instanceof> <r>obj</r> <type>java.awt.Point</type> </instanceof>
    …
</if>
```

This operator takes two arguments – an entity-reference element and a *type* element, and returns a boolean value.

4.7 Operator *new*

This operator is used to create new instances of Java classes, or to construct arrays. To create a new instance, a *type* element specifying the class is used as the first argument. Subsequent arguments

are used as needed for the constructor's arguments. Run-time overload resolution is used to select the constructor:

```
<var> <name>myFrame</name> <type>java.awt.Frame</type> </var>
    …
<set> <r>myFrame</r> <new> <type>java.awt.Frame</type>
    <String>My New Frame</String> </new> </set>
```

To construct an array, a *type* element specifying the component class is used as the first argument. The second, and subsequent – (if any), elements are integers specifying the size of successive dimensions of the array:

```
<var> <name>scores</name> <type>float[]</type> </var>
<var> <name>chessBoard</name> <type>byte[][]</type> </var>
    …
<set> <r>scores</r> <new> <type>float</type> <int>300</int> </new> </set>
<set> <r>chessBoard</r> <new> <type>byte[][]</type>
    <int>8</int> <int>8</int> </new> </set>
```

4.8 Boolean Operators

The operators, *and* and *or*, are used to combine the values of two boolean values. These are sometimes also called *short cut* operators because the second operand is evaluated only if required to determine the overall result.

The *not* operator inverts the value of its single boolean operand.

Java	XML	Example
&&	and	`<and> <lt> <r>ix</r> <int>32</int> </lt>` `<call> <name>allDone</name> </call> </and>`
||	or	`<or> <gt> <r>error</r> <float>0.01</float> </gt>` `<r>notAvailable</r> </or>`
!	not	`<not> <r>fileFound</r> </not>`

4.9 Bitwise Operators

The operators *band*, *bor* and *bxor* are used to combine the bits of their two integral operands. The *bnot* operator inverts the bits of its single integral operand.

Java	XML	Example
&	band	`<band> <r>status</r> <int>0x0c</int> </band>`
|	bor	`<bor> <r>pixels</r> <r>0xffff</r> </bor>`
^	bxor	`<or> <gt> <r>error</r> <float>0.01</float> </gt>` `<r>notAvailable</r> </or>`
~	bnot	`<bnot> <r>circPix</r> </bnot>`

4.10 Increment, Decrement and Negate

The *inc*, *dec* and *neg* operators specify an entity reference which is incremented, decremented or negated, respectively. The following table shows examples of their use. The *inc* and *dec* operators optionally accept a second argument that gives provides the value by which to change the value of the first argument.

Java	XML[4]	Example
++	inc	`<inc> <r>counter</r> <float>2.0</float> </inc>`
--	dec	`<dec> <m> <r>center</r> <name>x</name> </m> </dec>`
-	neg	`<neg> <x> <r>factor</r> <r>i</r> </x> </neg>`

5. Flow Control

5.1 *break* Statement

The *break* element causes execution to break out of a loop. This element must be empty (and must not contain any other elements or text.) andIt should preferably be used only in the form prescribed for empty elements in XML.

```
<while> <boolean>true</boolean>
  <block>
        …
    <!-- some iterative computation -->
        …
    <if> <lt> <r>error</r> <float>0.0001</float> </lt>
      <break/>
    </if>
  </block>
</while>
    …
  <!-- execution continues here if break is executed -->
```

5.2 *call* Statement

This element is used to call functions. Distinguished by their contents, Tthere are two kinds of call statements, distinguished by their contents, for calling Java methods (on objects and classes), and script functions respectively.

When used to invoke a Java method, the call element must contain other elements as follows:

- A reference to an object or the name of a class whose method is to be called. (Applicable only when invoking Java methods, and not when calling functions defined in the script.)
- The name of the Java method or script function to be called
- (A variable number of) Aarguments to the method or function[5]

The following example illustrates the invocation of methods on a Java object[6]:

```
<var> <name>myFrame</name> <type>java.awt.Frame</type> </var>
<var> <name>visible</name> <type>boolean</type> <value>true</value>
</var>
    …
```

[4] In Java the ++ and -- operators can be applied either as a prefix or a postfix. In this implementation we created an operator that only increments or decrements the named variable – but no value is returned.
[5] The type of arguments supplied is used to perform function overload resolution at run-time.
[6] The method *may* be a static member of its class. The method invocation illustrated is functionally equivalent to the following piece of Java: `myFrame.setVisible( visible );`

```
<call> <r>myFrame</r> <name>setVisible</name> <r>visible</r> </call>
```

The following example illustrates the invocation of methods on a Java class[7]:

```
<var> <name>x</name> <type>double</type> </var>
<var> <name>cos_x</name> <type>double</type> </var>
    …
<set> <r>cos_x</r> <call> <type>java.lang.Math</type> <name>cos</name>
<r>x</r> </call> </set>
```

The following example illustrates the use of call for invoking a function defined in the script (see section 2.4 above):

```
<var> <name>x</name> <type>float</type> </var>
<var> <name>fac_x</name> <type>float</type> </var>
    …
<set> <r>fac_x</r> <call> <name>factorial</name> <r>x</r> </call> </set>
```

5.3 continue Statement

The *continue* element is used within looping (*while*, *for*) constructs to skip execution of the remaining part of the (innermost) enclosing loop. This element must not contain any other elements or text, and should preferably be used only in the form prescribed for empty elements in XML.

```
<while> <boolean>true</boolean>
  <block>
        …
    <if> <call> <name>isDone</name> <r>serialNo</r> </call>
      <continue/>
    </if>
        …
        <!-- this part skipped if isDone returns true -->
        …
  </block>
</while>
```

5.4 *for* Statement

A *for* element is used to create a loop (iterative section of program) as in the following example:

```
<var> <name>rate</name> <type>float</type> </var>
<for>
  <set> <r>rate</r> <float>0.0</float> </set>
  <lt> <r>rate</r> <float>15.0</float> </lt>
  <inc> <r>rate</r> <float>2.0</float> </inc>
  <block>
      …
    <!-- compute and use compound interest -->
      …
  </block>
</for>
```

[7] The method *must* be a static member of the class. The method invocation illustrated is functionally equivalent to the following piece of Java: `cos_x = java.lang.Math.cos( x );`

The *for* element contains four statements – the first three exist for iteration control, while the last one is the body of the loop. The iteration control statements are analogous to those used with *for* statements in C, C++ or Java. The last statement is typically a block that contains all of the statements to be executed repeatedly.

5.5 *if* Statement

The *if* element is used to conditionally execute blocks of code, as in the following example:

```
<if> <lt> <r>rate</r> <float>0.0</float> </lt>
  <call> <name>lessThanZero</name> </call>
<elseif/> <le> <r>rate</r> <float>2.0</float> </le>
  <call> <name>betweenZeroAndTwo</name> </call>
<elseif/> <le> <r>rate</r> <float>4.0</float> </le>
  <call> <name>betweenTwoAndFour</name> </call>
      …
<elseif/> <le> <r>rate</r> <float>10.0</float> </le>
  <call> <name>betweenEightAndTen</name> </call>
<else/>
  <call> <name>greaterThanTen</name> </call>
</if>
```

In its simplest form, the *if* element contains a boolean expression and a statement. The statement being executed only if the boolean expression evaluates to true. An *if* can be augmented with any (0 or more) number of *elseif* elements (0 or more), and one optional *else* as in the example above. The *else*, when present, and each *elseif* must be followed by a boolean expression and a statement. In the general form illustrated above, the boolean expressions are evaluated in sequence till until one is found that yields *true* is found. The associated statement is then executed, and processing of the *if* statement is terminated.

Note that the *elseif* and *else* elements must be empty, and preferably be used in the form prescribed for empty XML elements. As usual, a block (containing statements and blocks) can be used as a conditional statement within an *if* structure.

5.6 *return* Statement

A *return* causes a function to return control to the caller optionally with a value (optional).

```
<return/>  <!-- permitted only in a 'void' function -->
<return> <r>status</r> </return>
```

5.7 *while* Statement

A *while* is another form of iterative statement that is controlled by a single expression, as shown in the following example:

```
<while> <gt> <r>error</r> <float>0.0001</float> </gt>
  <block>
      …
   <!-- iterate computation one more time -->
      …
  </block>
</while>
```

The boolean expression immediate following the start tag of the *while* is evaluated repeatedly, and the block within the body of the *while* is executed as long as the controlling expression evaluates to *true*. Execution on the *while* statement ends when the controlling expression evaluates to *false*.

6. Exceptions

6.1 *try* and *catch* Statement

A *try* element is the equivalent of Java's *try-catch* structure. An example *try* element is illustrated below:

```
<try>
  <block>
      …
   <!--  attempt operation that might throw exception here -->
      …
  </block>
<catch/> <name>e</name> <type>java.lang.Exception</type>
  <block>
      …
  <!-- handle e here -->
      …
  </block>
      …
<catch/> <name>se</name> <type>java.sql.SQLException</type>
  <block>
      …
  <!--  handle se here -->
      …
  </block>
</try>
```

A *try* element contains one statement (which may be a block), followed by any number of *catch* sections. Each catch section is parameterized with the type of exception it can handle. The statement (or block) associated with a catch is given control when an exception of the specified type occurs. The *catch* element serves to separate out the catch sections, and must be used in the form recommended for empty elements in XML.

6.2 *throw* Statement

A *throw* element can be used to throw an exception as in the example below:

```
<if> <gt> <r>error</r> <float>0.001</float> </gt>
  <throw> <new> <type>AppException</type>
      <String>Error-count threshold reached</String> </new> </throw>
</if>
```

Programming Languages in Testing

- C and C++
- Proprietary (equipment or vendor specific)
- Open Source

Embedded Scripting Languages

- Tcl (for C and C++)
- DynamicJava (for Java)
- Axess

Java and XML-Based Scripting

- XSLT
- XMLScript (simplified XSLT)

Design of AXESS

- Java Compatible
- Java-like Usage
- XML based
- Simple
- Embeddable
- Compact
- Efficient

Use of AXESS

- Standalone use
- Embedded within Java programs

Invoking Axess Standalone

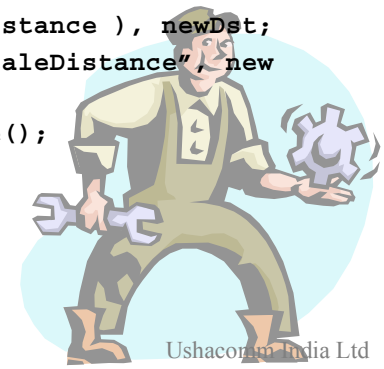
```
C:> java Axess scriptOne.xml
```

Example Standalone Script

```
<Axess>
    <function> <name>scaleDistance</name> <type>float</type>
      <arg> <name>oldDistance</name> <type>float</type> </arg>
      <block>
        <var> <name>newDistance</name> <type>float</type> </var>
        <!-- body of function -->
        <return> <r>newDistance</r> </return>
      </block>
   </function>

   <function> <name>main</name> <type>void</type>
      <arg> <name>args</name> <type>java.lang.String[]</type> </arg>
      <block>
      <!-- body of function -->
      </block>
   </function>
</Axess>
```

Embedded Use

```
Axess scr1 = new Axess( new java.io.File(
   "scriptOne.xml" ) );
float oldDistance, newDistance;
   …
Float oldDst = new Float( oldDistance ), newDst;
newDst = scr1.callFunction( "scaleDistance", new
   Object[] {oldDst} );
newDistance = newDst.floatValue();
   …
```



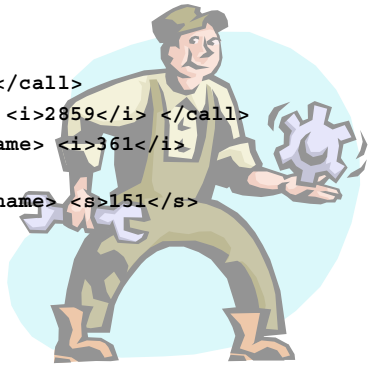
Applications of AXESS

- GUI-based testing tool
- Messaging with VoIP device

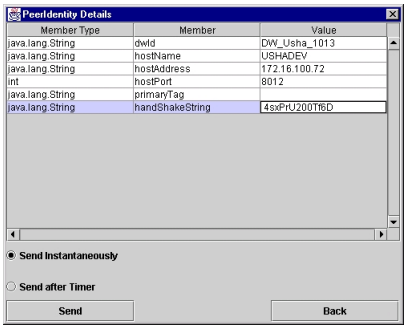
Application: GUI-Based Testing

- Java tool captures GUI events
- Events are replayed at later time

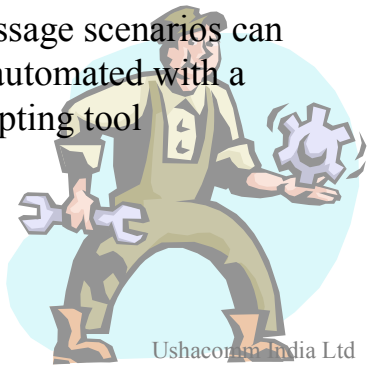
```
<call> <name>auto.RoboUser.init</name> </call>
<call> <name>auto.RoboUser.delay</name> <i>2859</i> </call>
<call> <name>auto.RoboUser.clickOnXY</name> <i>361</i>
   <i>64</i> </call>
<call> <name>auto.RoboUser.typeInText</name> <s>151</s>
   </call>
```



Application: Messaging with VoIP Device



- Manual procedure required use of a GUI to create each test
- Message scenarios can be automated with a scripting tool

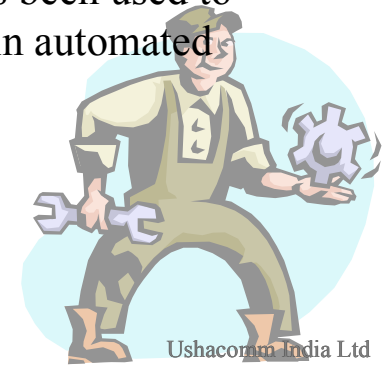


Conclusions I

- XML is an optimal method to represent and store structured information; it must, therefore, be applicable to programs as well.

Conclusions II

- A XML extension has been designed to represent any Java program. This programming language has been used to represent test scripts used in automated testing.

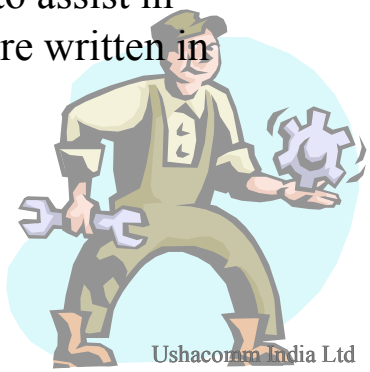


Conclusions III

- A disadvantage of programming languages based in XML is verbosity. This is particularly apparent when representing Java. However, the use of XML enables a tools-based approach, and it will be possible to use formal techniques to prove properties of testing scripts.

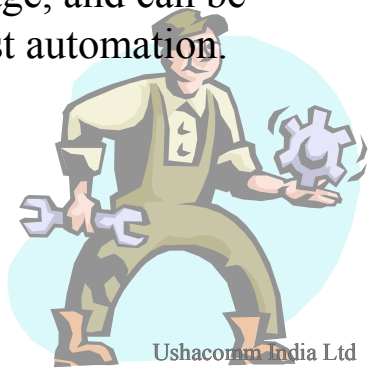
Conclusions IV

- A compact and efficient Java-based interpreter for XML programming scripts has been created, and used to assist in automated testing of software written in Java.



Conclusions V

- The ability of the interpreter to integrate with test environments and software written in Java is a definite advantage, and can be used to raise the level of test automation.



QWE2000 Session 12A

Dr. Adam Kolawa [USA]
(ParaSoft)

"Testing Dynamic Web Sites (12A)"

Key Points

- Testing techniques for Dynamic Web sites
- Effectively and Efficiently testing Web applications

Presentation Abstract

Today's dynamic Web sites are sophisticated n-tier software applications with Web interfaces. Because dynamic Web sites involve extensive programming, developers of these Web sites need to apply their standard testing procedures in their Web development. In this session, we will discuss how to use the following techniques for testing Web applications.

--Using White box testing to test the site's construction. White box testing in software development involves calling every method or class. In Web development, this involves testing every static and dynamic page for coding errors and li nk errors.

--Using Black box testing to test the site's functionality. Black-box testing helps ensure that a program the way it was intended to function. In dynamic Web sites, the states of pages are constantly changing depending on how a user visits a site.

--Using Web-box testing, a method of testing one dynamic page at a time, to test at the program level as well as the output level (script and HTML page). Using Web-box testing, will help you expose core dumps, uncaught exceptions, or any ot her problems with code.

--Using Regression testing to uncover errors made during code modifications. Developers use regression testing to help ensure any changes made to code does not cause any other errors. In Web development, regression testing also helps ensure that code changes do not cause any other problems.

Attendees will learn the following from the presentation:

--How Web developers and software development are similar
--Software development practices that Web developers can borrow
--Techniques for effectively and efficiently testing Web site pages

--Techniques for effectively and efficiently testing software programs that generate dynamic pages

About the Speaker

Adam Kolawa is the CEO of ParaSoft Corporation, a leading provider of software productivity solutions. Dr. Kolawa holds a Ph.D. in theoretical physics from the California Institute of Technology. Dr. Kolawa has extensive experience in both programming and managing programmers. He has written several successful commercial software products, and has headed the development of a large number and variety of software projects, including software development tools, retail solutions, Web development and management tools, data mining tools and more. Heading a company that sells products to programmers and managers has given Dr. Kolawa the unique opportunity to interact with software developers and managers at mutiple levels. Dr. Kolawa established his ability to clearly communicate the expertise he has gained from his education and experience in the many papers on physics and computer science that appeared in such publications as the Caltech Concurrent Computing Project Memo and the Proceedings of the Hypercube Conference. Two of Dr. Kolawa's papers are included in Parallel Computing Works!. Ed. Geoffrey Fox. Dr. Kolawa has spoken at many conferences including STAR East '99, JavaOne, Linux Expo, Sigs Conference for Java, Software Development East, SGI Expo, Quality Week, IEEE conferences and has also conducted national seminars on Strategies for Effective Software Development.

ParaSoft

Testing Dynamic Web Sites

Strategies for Applying Traditional Software Testing Techniques to N-Tier Web Applications

ParaSoft

USA France Germany UK

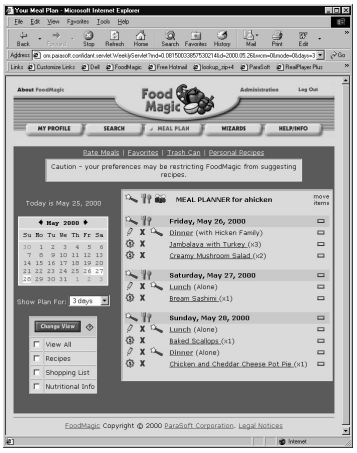
In the Beginning...

was the Web,
and the Web
was *static*.

ParaSoft

Dynamic Sites Appear

- Content on demand
- Database connection
- Web browsers as simple GUIs



ParaSoft

3

Today's Dynamic Web Sites

- Complex n-tier software applications
- Theoretically contain an infinite number of pages
- Require sophisticated testing

ParaSoft

4

Static vs. Dynamic

- **Static** pages exist as actual files
- **Dynamic** pages are created on-the-fly by back-end applications (CGI, servlets, scripts, etc.)

ParaSoft

5

Testing Static Sites

- Relatively straightforward task
  - Pages are “real” and have only one instance
  - Tools can automate many aspects of the process
- Task includes:
  - Checking links
  - Checking anchors
  - Checking spelling
  - Validating HTML code
  - Finding orphaned files

ParaSoft

6

Testing Dynamic Sites

- Proven software testing techniques apply
  - Coding Standards
  - Automated Tools
  - Error Prevention
  - Error Detection
- Test individual elements/components
- Test the whole site
- Verify

ParaSoft

7

Which Techniques Apply to the Web?

- White-box (structural) testing
- Black-box (functionality) testing
- Regression testing
- Coding standards enforcement
- Web-box testing (specialized process of testing one dynamic page at a time)

ParaSoft

8

Typical Web Site Errors

- Servlets that throw exceptions
- Multiple types of broken links
- CGIs that core dump
- Databases that crash
- Various HTML, Cascading Style Sheet, and JavaScript errors

ParaSoft

9

White-Box Testing

In traditional software development:

- Determines whether application will crash
- Checks application's construction
- White-box testing means slamming doors and kicking tires

In dynamic Web sites:

- Create instances of all static *and* dynamic pages
- Ensures all back-end applications are robust
- Verifies that the program created valid HTML

ParaSoft

10

White-Box Testing

Testing at the program level:

- Trigger each page
  - load static pages
  - execute programs with a set of inputs
  - view pages that the programs return
- Create baseline for future stages of testing

Testing HTML construction and output:

- Coding standards are especially critical in HTML
  - No compiler to catch errors
  - Checking page display in a browser is unreliable
- HTML coding standards prevent errors

ParaSoft

11

What Can Standards do?

An ounce of prevention…

- Data input errors that corrupt data
- Dynamic content errors
- Errors that affect localization
- Navigation-related errors (not “traditional” link checking)
- Performance issues
- Browser incompatibilities
- Incorrect page appearance
- Security
- Accessibility

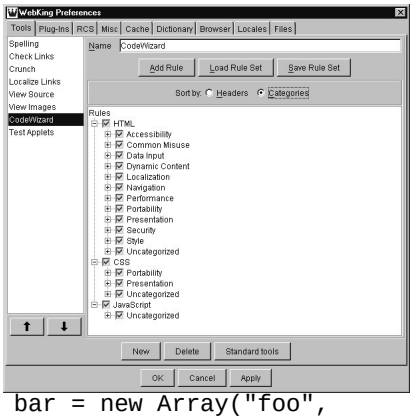
ParaSoft

12

Coding Standard Violations

- **Missing content:** `Contact us at <B> </B>. <IGM SRC="picture.gif">`
- **Navigation:** `<A HREF="index.htm">`
- **Dynamic content errors:**
  For <SCRIPT>,
  TYPE or LANGUAGE
  attribute must be set.
- **Presentation problems:** `H1 {font-size:14s }`
- **Portability Problems:**

```
function foo() { var bar = new Array("foo",
"bar"); bar.pop(); }
```

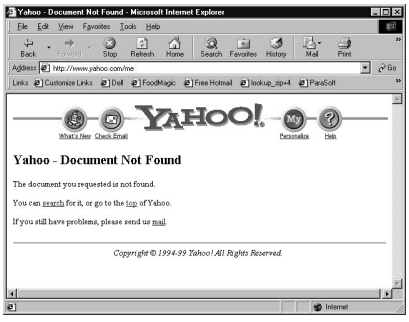


ParaSoft

13

Checking Links

- Part of white-box testing is to perform thorough link checking
  - Malformed URLs
  - Broken anchors
  - Missing pages
  - Empty URLs
  - Duplicate anchors



ParaSoft

Link Errors Gotchas

1. Broken anchors

- Most link checkers do not test whether links to anchors (<A NAME="...">) actually lead to the correct anchor
- Tools that do check usually just verify that the link leads to a valid page

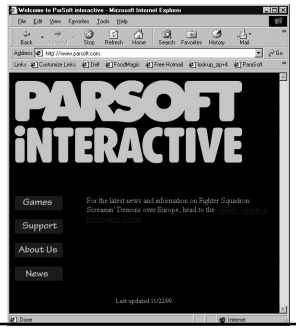
ParaSoft

15

Elusive Link Errors

2. Malformed URLs

- Most link testing tools restrict testing to server-side links that have a valid <A HREF=" "> statement, and fail to recognize any other links
- These links do not necessarily result in delivery of the requested page
- It is also important to test for broken client-side links
  - http://www.parasoft.com
  - http:/www.parasoft.com
  - http//www.parasoft.com
  - http://www.parsoft.com



ParaSoft

Black-Box Testing

- Tests whether an application functions as intended
- Compares actual functionality with intended functionality described in specifications
- Two relevant types of black-box testing:
  - Critical paths' functionality
  - Verifying existence of certain invariable elements

ParaSoft

17

Testing Critical Paths' Functionality

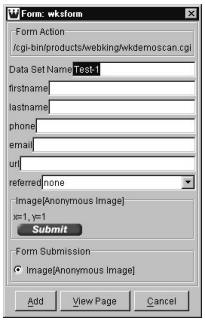
- Two main difficulties of creating exact instances of dynamic pages:
  - Passing the appropriate parameters
  - Setting the program's internal variables

ParaSoft

18

Testing Critical Paths' Functionality

Example: Testing credit card verification



- Program Input:
  - Payment type
  - Credit card type
  - Credit card number
- Internal variables
  - User name
  - Password
  - Items in the shopping cart
- Correct inputs without correct variables, creates a different page

ParaSoft

19

Checking for Invariable Elements

Common invariable elements include:

- Navigation bars
- Menus
- Clocks
- Calendars
- Advertising banners
- Buttons
- Copyright and contact information

ParaSoft

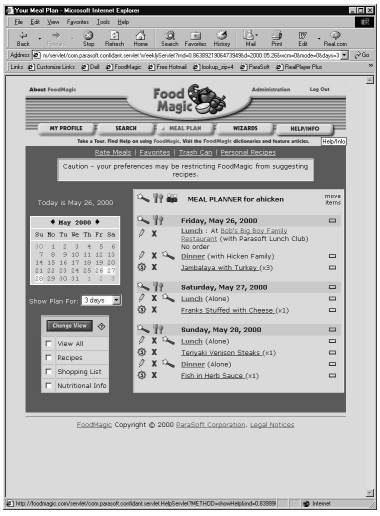
20

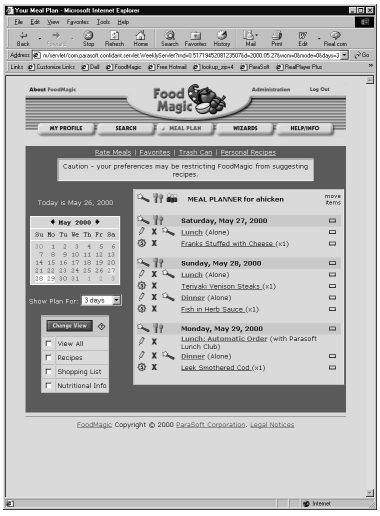
Checking for Invariable Elements

- Possible solution: Use a tool to create a test case. IE record/playback
- Obvious complication: False positives. IE calendars

ParaSoft

Invariable Element Example





ParaSoft

Checking for Invariable Elements

- Describe content, design, or presentation features that you want to appear in specified pages
- Describe page features in an invariant way
- Avoid false positives, only real true errors
- Language is purposely ambiguous

ParaSoft

Regression Testing

- Tests whether changes have introduced errors
- Tests whether your site continues to pass the white-box and black-box tests it passed previously
- Should be a simple process if you save all white-box and black-box test cases each time you test

ParaSoft

Web-Box Testing

- Modeled after traditional unit testing
- Adapted to accommodate dynamic Web sites
- Tests at the program and output level as early as possible
- "Test as you go" approach prevents errors
- Exposes exceptions, core dumps, and other problems in back-end applications
  - Verifies that HTML output pages are correct
  - Checks functionality of output pages (links, spelling, invariant elements, etc.)

ParaSoft

25

Web-Box Testing

1. Establish an infrastructure for building, publishing, and testing programs and scripts
  - Use a `makefile` to automate "housekeeping" functions such as:
    - Compiling programs
    - Transferring files to the correct directory of the Web server
    - Modifying databases
    - Initializing objects
  - Use publishing infrastructure to deploy programs automatically each time you modify

ParaSoft

26

Web-Box Testing

2. Set up tool checks for programs and scripts
- Automatically enforce coding standards for languages in which applications were written
- Perform automatic unit testing for C/C++ and Java applications
- Use a runtime error detection tool to check for memory corruption in C/C++: especially important for applications that work with CORBA and other object brokers

If back-end applications do not pass tool checks, publishing of your site should be canceled

ParaSoft

27

Web-Box Testing

3. Enforce HTML coding standards
- Use W3C standards

4. Inspect the output pages
- Trigger specific instances of each page by setting internal state variables
- Look for broken links, broken anchors, spelling errors, etc.
- Check for required page elements

ParaSoft

28

Summary

- Tools for Building/publishing
- Traditional program error detection
- Prevention is **much** cheaper
- Static HTML analysis
- Coding Standards
- Thorough Link checking
- Regression testing
- Record/Playback & Scripting
- Sensible Server Response

ParaSoft

29

Conclusion

- Dynamic Web sites are bringing traditional developers into Web development.
- Now is the time apply traditional proven development testing techniques to the Web.
- Integrating these techniques into the Web development cycle create the high quality Web applications while slashing development time, effort, and cost.

ParaSoft

30

Testing Dynamic Web Sites

Dr. Adam Kolawa, ParaSoft Corporation

Introduction

Not long ago, Web sites were collections of simple, static pages that merely displayed information in the form of unchanging text and images. But today's dynamic Web sites are sophisticated n-tier software applications with Web interfaces. Software developers who are now building these dynamic Web sites typically want to apply their standard development procedures to their Web development projects.

Most of all, developers need a way to prevent and detect errors as they develop n-tier Web applications. Testing techniques that have proven to be successful in conventional development can also apply to the Web. Web developers should test all elements of their dynamic site, then test the whole site together. It is important to verify that each piece of the site will behave as it should. One way to accomplish this is to perform white-box testing, black-box testing, regression testing, coding standards enforcement, and unit testing throughout the site building process.

By adapting and implementing these traditional testing techniques, Web developers can:

- Create an instance of every static page (a page that exists as an actual HTML file) and dynamic page (a page that is returned by a form-processing program such as a CGI, servlet, or script) available on a site
- Map the various paths through a site
- Test a site's construction (white-box testing)
- Test a site's functionality (black-box testing)
- Maintain site integrity (regression testing)
- Test one dynamic page at a time

Applying conventional testing techniques helps developers create the highest quality dynamic Web sites possible. At the same time, it reduces the time and effort required for development and testing. We will now explain how each of these testing concepts can be applied to dynamic Web site development.

White-Box Testing

By now, most developers are familiar with the concept of white-box testing. White-box testing determines whether any parts of the application will crash when the application is used in expected or unexpected ways. During white-box testing, you can check an application's construction by examining the source code and then creating and executing tests designed to flush out errors. We can draw an analogy between white-box testing and inspecting a new building. White-box testing

for a new building might involve testing every building element that you encounter--including walls, doors, windows, and doorknobs--by pounding, slamming, twisting, or kicking it. Performing white-box testing on a dynamic Web site requires us to create an instance of every static and dynamic page, then test each page in ways that expose construction errors.

There are two levels to performing white-box testing on a dynamic Web site. First, we must ensure that all of our back-end applications that generate pages are built with robust code. For example, these back-end applications must not contain exceptions and must not core dump. Second, we must verify that the program actually created the HTML pages we wanted.

To begin white-box testing, we first must examine a site's programs and static pages to determine the most effective way to test the site. Next, we must trigger each page by loading static pages, executing programs with a set of inputs, then viewing the pages that the programs return. Finally, we must make note of our site's file and directory structure. Having gathered this information, we know what to expect during future stages of testing.

After triggering each page, we must thoroughly test whether each page's links, including links to and from dynamic pages, will operate properly. We must also check all HTML, JavaScript, and Cascading Style Sheet (CSS) code against coding standards to be sure that the pages will display as we intended. One of the greatest problems of testing dynamic sites is figuring out how to test pages that don't exist as physical HTML pages. After we trigger each page, we can apply the most sophisticated tests available to the pages, with the intention of exposing a large number and variety of errors. The types of problems we should be looking for include:

- Servlets that throw exceptions
- CGIs that core dump
- Databases that crash
- Multiple types of broken links
- HTML, CSS, and JavaScript problems, including:
    - Data input errors that corrupt data between the user form and the program on the server.
    - Dynamic content errors that deliver inconsistent results from the program or script that generated them to the user.
    - Errors that affect localization (for those doing business across the globe in a wide array of languages).
    - Navigation-related errors that prevent visitors from reaching the place they thought the link led to. (It is important to go beyond the errors normally reported by traditional "link-checkers" by recognizing malformed URLs and testing them as links).
    - Performance issues that can slow down your Web site.
    - Browser incompatibilities, portability issues, and old tags from earlier HTML standards that may cease to be supported.
    - Presentation errors that can affect the way your page looks.
    - Security errors that affect data protection for those relying on client/server certificates to prove identity.

- Style issues that can make the code more error-prone.
- Accessibility issues for users with special needs.

Here we come to a change of mindset for developers who are moving from conventional software to the Web. Developers are not used to structured inputs to their applications. HTML is the first type of input that can really be considered a structured input. Developers can apply a number of different algorithms to check it. HTML also gives a structured output, in which developers should check links, spelling, and code for common errors.

By using white-box testing techniques to test your site based on its code structure, you overcome the main problem inherent in testing frequently-changing sites: every time you change the site, you need to create new tests. If you generate and execute the same tests each time that you test the site, you can test a dramatically changed site with about the same effort that you use to test a slightly-modified site.

Black-Box Testing

Black-box testing tests whether an application actually functions as it is intended to function. You can perform black-box testing by comparing an application's actual functionality with the intended functionality described in the application's specification document. To use the building inspection analogy again, black-box testing might involve checking that every building element included in the blueprint is actually present and meets the specifications. There are two types of black-box testing we can apply to dynamic Web applications:

- Testing critical paths' functionality (i.e., checking certain functionality by testing if associated paths through the site contain errors).
- Testing whether all appropriate pages contain certain invariable elements.

Testing critical paths' functionality

The problem with testing critical paths through the site is creating the exact pages that you want to test. There are two main difficulties involved in creating these exact pages:

- Passing the appropriate parameters to the program that creates each dynamic page (i.e., entering appropriate form inputs).
- Setting the program's internal variables to the values they would be set to if a user had actually taken the path through the site that you are trying to test.

For example, let's assume that you want to test your site's credit card processing functionality. In order to test this functionality, you would not only have to give the program inputs that indicate payment type, credit card type, and credit card number, but you would also have to set the program's internal variables in such a way that the program has a user name and password, and has some items in the shopping cart. If you created this program's return page by the submitting the correct inputs but not the correct variables, you would get a dramatically different version of the page, and would not be able to accurately test this aspect of the site's functionality.

Fortunately, you can use the information you gathered during white-box testing to help you here. During white-box testing, you mapped a default set of paths through your site. Now you can examine the test cases that you used in white-box testing, then extend these test cases or build new test cases to indicate exactly which paths through the site you want to test. As you add more inputs into forms, you will be able to navigate through more and more dynamic pages in whatever paths you choose to follow.

For example, let's assume that you wanted to test functionality on your site, which allows users to receive daily news stories that are tailored to the interests they specify. To test this functionality, you would add inputs and expand default paths until your path tree represented all possible paths that you were interested in testing. Then you would run your full array of tests on those paths to expose any errors related to the site's functionality.

Testing to ensure all dynamic pages contain certain invariable elements

There is a second facet to performing black-box testing on a Web site. We must ensure that specific items required by the page or application's specifications actually appear in the appropriate pages.

For example, pretend again for a moment that you have a site that delivers specially selected news stories to users. Your specifications say that every page titled "My News" needs to contain a calendar with the current date highlighted. How would you test that this functionality was indeed implemented? If you had a tool that performed functional testing at the GUI level, you could try to create a test script that played that test and alerted you to any change in that graphical element. However, if your site was working correctly, the calendar would change every day, and an "error" would be reported for any calendar that did not match the one in your control case. Thus, you would get false errors ("false positives") every day except for the day that you created the test

When you are working with a frequently changing application (as most Web developers are), creating rules that enforce specifications is considerably more efficient than checking specifications by graphically indicating and testing whether certain pages contain certain elements. To this end, we have designed our own language that we can use to describe content, design, or presentation features that we want to appear in specified pages. During white-box, black-box, and regression testing, we enforce the "rules" we create with this language. Because this language lets us describe these features in an invariant way, we avoid false positives (as are reported in the above example) and only worry about true errors. For example, we can enforce the calendar functionality described above by writing a rule that specifies that a certain type of page contain a certain type of calendar with dates that fall within a certain valid range.

Regression Testing

Regression testing tests whether changes have introduced errors into an application. To return one last time to the building inspection analogy, regression testing might involve performing all previous white-box tests (pounding walls, slamming doors, kicking railings on the stairway) and black-box tests (making sure shutters are still attached in the right locations according to blueprint spec-

ifications) to determine if a natural disaster affected the building's integrity. In terms of a Web application, regression testing would test whether your site continues to pass the white-box and black-box tests that it passed previously.

Performing regression testing should be simple if you save all white-box and black-box test cases each time you test. When you make new modifications, you can execute all previous white-box and black-box test cases, which should alert you to any new errors.

Specialized Unit Testing

There are additional unit testing techniques that are unique to dynamic Web sites, such as the process of testing one dynamic page at time. This process is modelled after traditional unit testing, but is adapted to accommodate the complexities unique to dynamic Web sites. The goal here is for dynamic Web site developers test their programs or scripts at two levels--the program level and the output level--as early as possible. The precise meaning of "testing at the program level" depends upon the language in which the program was written. For example, it could mean exposing all exceptions (Java), exposing core dumps (C/C++), exposing other programming failures (in Visual Basic and other languages), or making sure that scripts do not exit. "Testing at the output level" entails testing that the HTML page returned by the program is actually correct. By testing at both of these levels, you can expose problems associated with all aspects of the program, and fix these problems at the stage when it is easiest and fastest to do so.

The first step in establishing an efficient testing process is establishing the infrastructure for building, publishing, and testing the program or script. In most cases, you can set up this infrastructure using a makefile that allows you to automate many of the housekeeping functions related to program creation and publication. These housekeeping functions might include:

- Compiling programs
- Transferring files to the correct directory of your Web server
- Modifying databases
- Initializing objects

This publishing infrastructure can and should be used to automatically deploy your programs each time that they are modified. You can make this deployment process even more efficient by incorporating special tool checks into it. If you have testing tools you use to monitor the quality of your pages, you can automatically run appropriate files through specific tests each time that you try to modify your Web site. If these tools detect any problems, your modifications will not be published.

For example, if you have Java programs on your site, you might want to use a Java testing tool to check automatically during publishing, so that you can ensure that your Java programs follow coding standards and pass unit testing. If you are developing in C/C++ and are concerned about memory corruption (which is a particularly critical issue when applications work with CORBA or other object brokers), you would want to use a runtime error detection tool.

The second level of testing is performed by inspecting the page that the program or script returns. To do this, you need to trigger specific instances of this page to see what the program or script really did. Dynamic pages normally cannot be triggered on their own, so you need to have internal state variables set. For example, to test a shopping cart page on your site, you would need to set the program's internal variables so that the shopping cart page contained several items. You can either set variables by hand or by clicking through the prerequisite pages. These solutions may seem time-consuming and tedious at first, but they are essential to thorough site testing. After you have created and tested the return page, you can manually examine the page to determine whether the program or script actually does what it is supposed to do. You can also test this page for HTML errors, link errors, CSS errors, and JavaScript errors, and you can ensure that the page contains certain invariant elements by enforcing any specification rules you have created.

Coding Standards Enforcement

Coding standards are language-specific "rules" that, if followed, significantly reduce the opportunity for making errors. In order to maintain quality Web applications, managers should implement and enforce coding standards in every programming language, for every developer, on every project. If they do so, they can prevent errors from ever entering the code. Enforcing coding standards will result in higher quality code—in less time—than would be possible by allowing developers to make errors and having them perform extensive debugging to find and fix their errors.

You can apply HTML, JavaScript, and CSS coding standards to both static and dynamic pages as you perform white-box, black-box, regression, and Web-box testing. This helps you uncover coding problems in your static pages, and—more importantly—helps you target what programs have the potential to generate a virtually unlimited number of dynamic pages with poorly constructed HTML, JavaScript, and CSS code.

By tailoring coding standards to your team or project's exact needs, you can ensure that you'll find the precise problems that you are trying to target and will not have to deal with irrelevant errors. By building a custom coding standard each time that you find a coding error, you can rapidly construct a set of coding standards that will prevent your most common coding mistakes from recurring.

HTML coding standards address a wide range of issues. Some standards address simple yet harmful style errors such as providing empty tags in a document. For example, perhaps you wrote the tags <B> and </B> but did not put any text between them. If you intended to display an important piece of information in boldface but neglected to provide the text, the contents of your page will be incomplete. Coding standards can also alert you when you have written a tag that has no effect.

Of course, coding standards also prevent many errors that have a more serious effect on a site. Browser incompatibility is a thorn in the side of many a Web developer. Certain tags, attributes, and attribute values display properly in one browser only to be unrecognized in another. For example, developers using the ALIGN attribute within the <IMG> tag should be aware that only the values `top`, `middle`, `bottom`, `left`, and `right` will display properly in both Netscape Navigator and Internet Explorer. To maintain the integrity of each page's display, it is wisest to use only

these universally-accepted attributes. Non-standard attributes such as `absmiddle`, `baseline`, and `absbottom` are recognized only by Netscape.

HTML coding errors can also harm your site's accessibility. Consider the `accesskey` attribute within a form. You use this attribute to create a shortcut key, which will allow users to jump to a particular element in a form with a few keystrokes. If you fail to specify a correct value for this attribute, you will hinder your users' ability to navigate through your document. The `accesskey` attribute is not yet widely used, but it will become increasingly important for Web developers who want to offer the ultimate level of convenience to their visitors.

Of particular interest to dynamic Web site developers is the coding standard that for each <SCRIPT> tag, a TYPE or LANGUAGE attribute must be set. If you are using JavaScript or Visual Basic in your HTML pages, you must provide this information in the TYPE or LANGUAGE attribute to ensure that the server will handle your embedded scripts properly. It is worth noting that LANGUAGE will be deprecated in HTML 4.0. TYPE will become the new standard.

Conclusion

The advent of dynamic Web sites has brought traditional developers into Web development. These developers are accustomed to applying certain types of testing techniques throughout their development process, and would like to do the same with their Web development, but they sometimes lack the knowledge to translate the techniques to the Web. Now is the time to take traditional development testing techniques and apply them to dynamic Web sites. By integrating these techniques into your Web development cycle, you will be able to create the highest possible quality Web applications while slashing development time, effort, and cost.

QWE2000 Session 12I

Mr. Steven D. Porter [USA]
(Practical Consulting Group)

"From Web Site To Web App: Ensuring Quality In A Complex Environment"

Key Points

- Differences between websites and web applications
- Things to consider for a Quality Strategy
- Some metrics as a guideline for estimating time needed for testing

Presentation Abstract

The continued success of web development teams will depend heavily on the ability to make the transition from deliverying web sites to delivering reliable, stable web applications.

We must develop a cost-effective strategy for quality assurance and testing through this transition and beyond. The foundation of this strategy must rest on a thorough understanding of the nature and dynamics of applications within a web environment. The key to success is separating the volatile aspects of the application and the web from the non-volatile ones, and focusing on the quality assurance efforts accordingly.

About the Speaker

Steven Porter has been a certified instructor for Rational Team Test since 1996. He has written a one-day Quality Assurance class that he teaches as part of the Object Oriented Project Management Certification Series developed by Advanced Programming Institute of El Dorado Hills, CA. as well as providing content for the Process Management course of the same series.

In 1998, he provided QA Management for a yearlong Y2K object-oriented project involving over 40 reengineered applications for CalPERS (California Public Employees Retirement System). Since leaving Advanced Programming Institute, he has consulted as lead tester for iSeer of Seattle, WA - a web development company, and is currently serving as QA Manager for Practical Consulting Group in Rancho Cordova, CA.

From Web Site to Web App: Ensuring Quality in a Complex Environment

sporter@2xtreme.net

1

The Medium is The Message

-Marshall McCluhan

- The Web Creates Great Potential
  - Increased Profits from e-Commerce
  - Lower Barriers of Entry
  - Easier Information Distribution and Access
- The Web is Developing at Warp Speed
- The Web is Still Immature, But is Being Asked to Perform in a Mature Manner

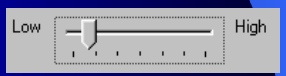
2

Agenda

- "Web Site" vs. "Web Application"
- The Web Paradox & Its Effect On Quality
- Forces that Affect Quality
- Quality Strategy Overview
- Examples
- Wrap Up

Web Site

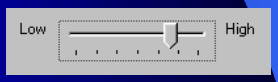
- Static Pages Built with HTML
- Organized Documents and Files Accessed Through Hyperlinks
- HTTP Protocol
- Site Accessed Through Browsers
- Capable of Providing Secured Access
- Examples
  - www.iseer.com
  - www.a-p-i.com



Volatility
Complexity
Risk

Web Application

- Contains the Elements of a Web Site and Extends the Capabilities to:
  - Allow User Input
  - Perform Standard Database Transactions
  - Perform Business Logic
  - Provide Client and Server Side Validation of Data
  - Perform Queries and Display Dynamically Generated Reports
- Examples
  - On-line Reservation Systems
    - www. AlaskaAir.com
  - On-line Sales
    - www.Amazon.com



Volatility
Complexity
Risk

5

The Web Paradox

- Web Browsers allow for Platform Independence
  - Thin Client / Fat Server
- Competing Web Browsers do not:
  - Uniformly Display Content
  - Uniformly Perform Logic
  - Uniformly Accept User Input
- Examples
  - Buttons Do Not React the Same in Netscape and Internet Explorer.
  - Some Browser Versions Do Not Support JavaScript
  - File Downloading is Handled Differently from Browser to Browser.
- User Plug-Ins Effect Capabilities

6

Effect on the Quality Strategy

- The Strategy for Quality is becoming more Complex due to the Rising VCR Forces.
  - The Web is a Cauldron of Rapid Change
  - The Transition from Building Web Sites to Web Applications Increases Complexity and Risk
  - Escalating User Expectations Increases Complexity
  - Security and Privacy Issues Increase Risk
  - Market Demands Compress Development Schedules

7

Quality Strategy

- Organize the Quality Effort into Categories
- Establish the Standards of Quality for each Category
- Manage the VCR Forces that Influence Quality
  - Volatility
  - Complexity
  - Risk
- Negotiate Cost and Time
- Implement the Strategy
- Manage and Leverage Resources to Accomplish our Purposes

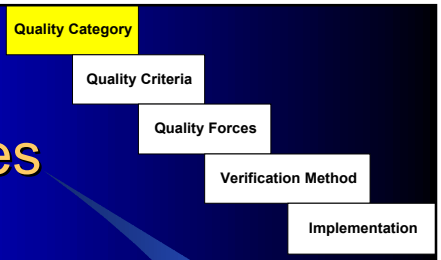
8

Strategic Elements

What are the basic categories that relate to quality?

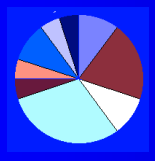
9

Quality Categories



Examples

- Theme
- Content
- Presentation
- Navigation
- Data Exchange
- Performance
- Technology
- Security



10


Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Create Standards
Product
– Based on Quality to the User
– Based on Quality to the Business
– Based on Technical Constraints
Copyright © 2000 by Steven D. Porter
11


Quality to User
Responsiveness
Accuracy
Completeness
Ease of Use
Relevance
Aesthetics
Security
Privacy
Reliability
Copyright © 2000 by Steven D. Porter
12

Quality to Business

- Customer Impact
- Sales
- Business Credibility

- Cost
- Time to Delivery
- Reliability / Maintenance

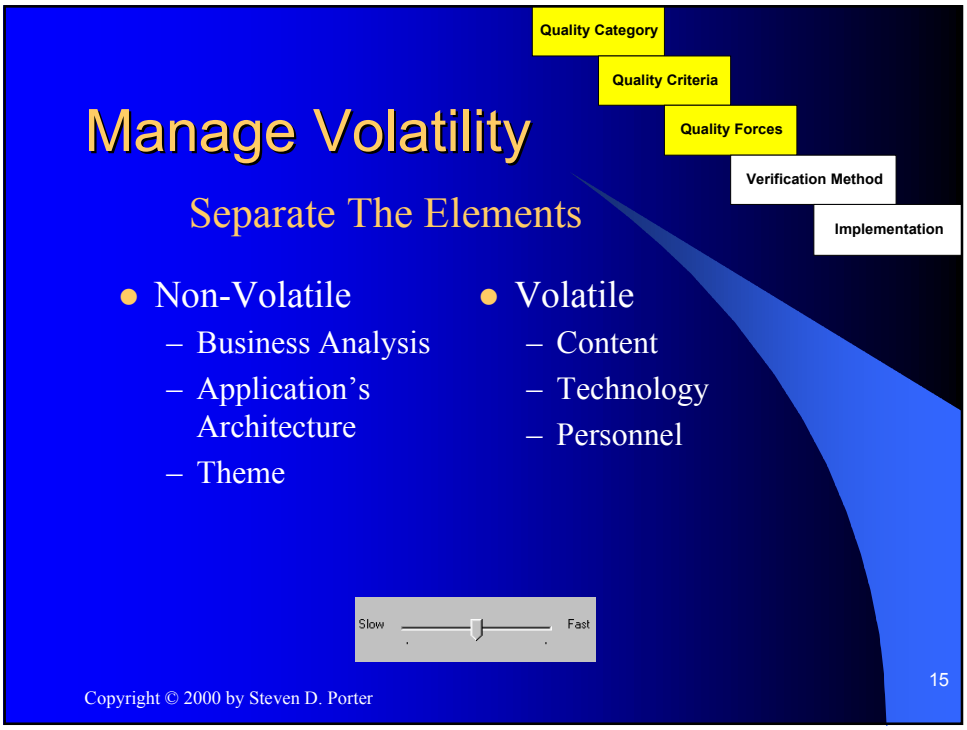
13

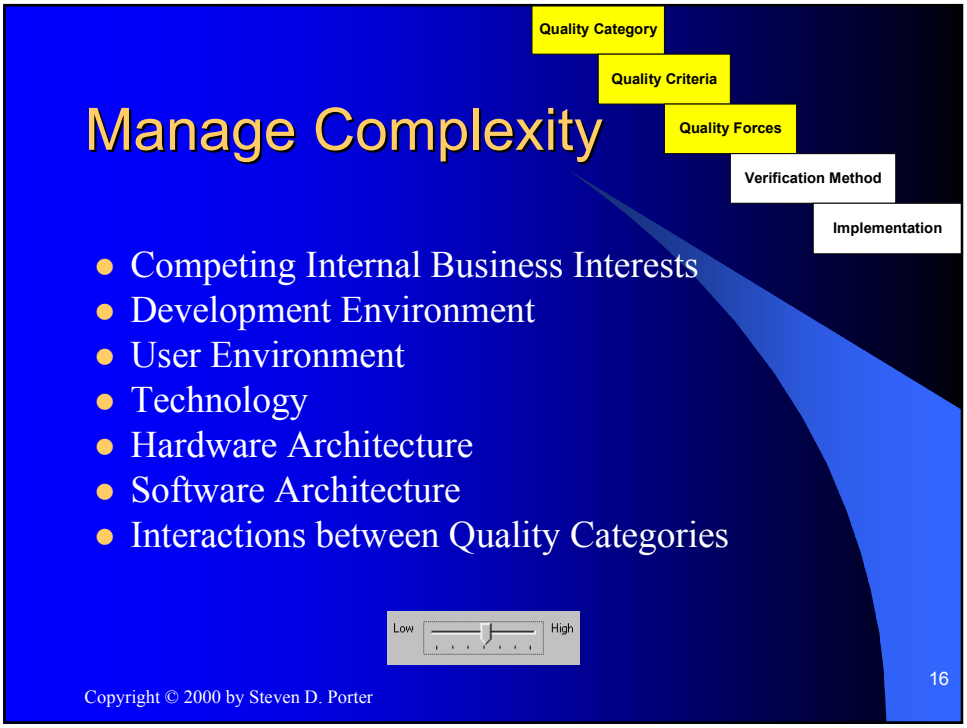
Create Standards

Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation

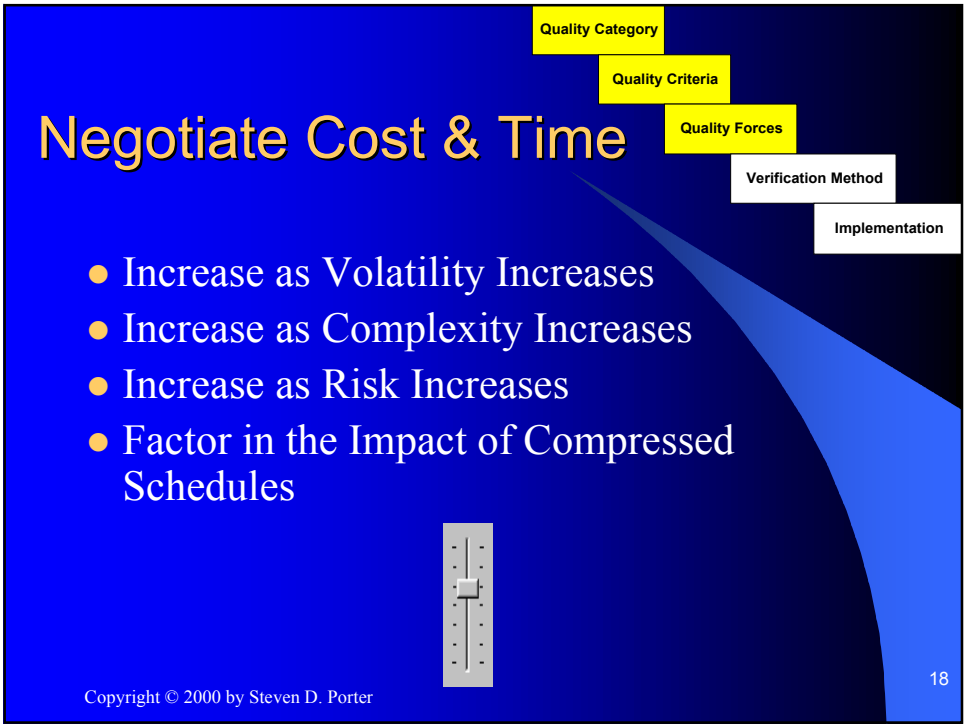
- Development Process
  - Input Artifacts
  - Activities and Tasks
  - Output Artifacts
  - Change Management
  - Roles and their Relationship to other Roles
- Manage to the Standards

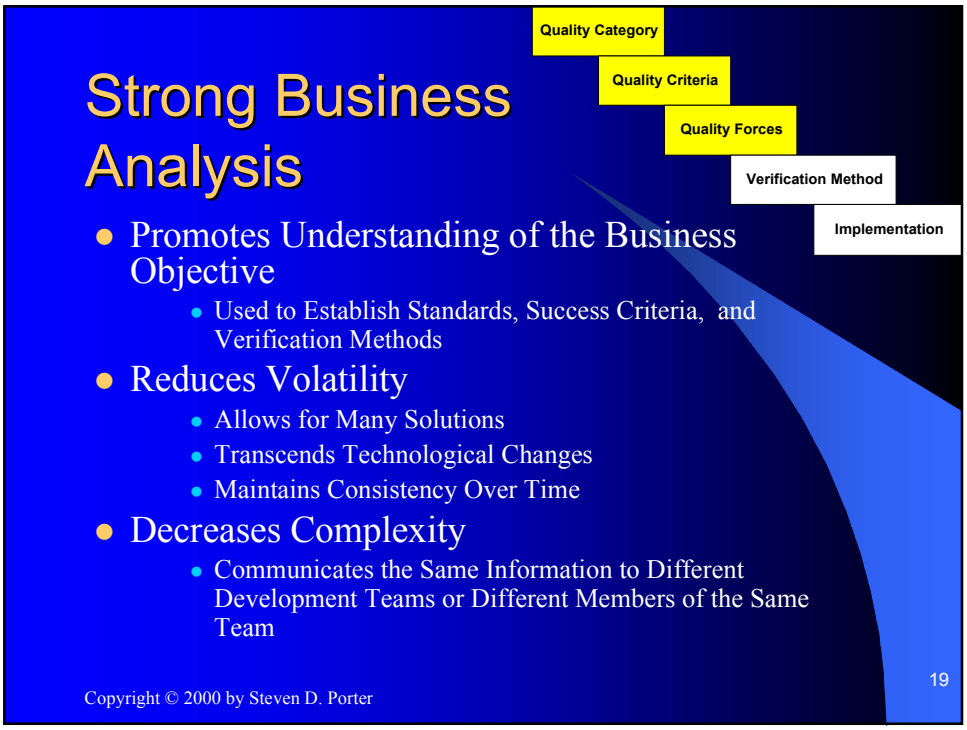
14

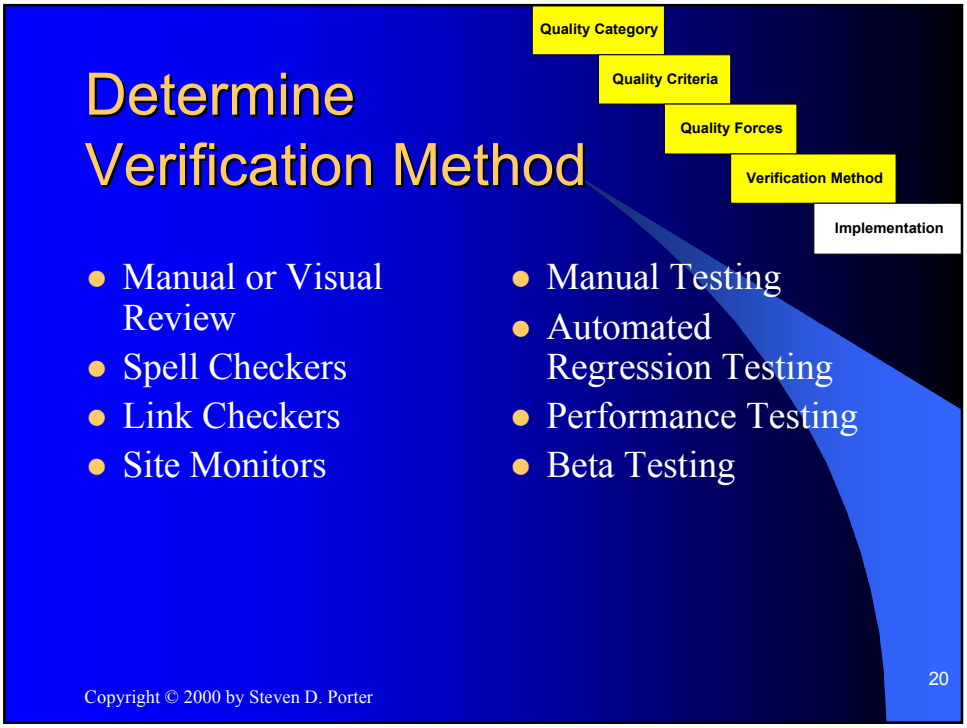

Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Manage Volatility
Separate The Elements
Non-Volatile
Business Analysis
Application's Architecture
Theme
Volatile
Content
Technology
Personnel
Slow
Fast
Copyright © 2000 by Steven D. Porter
15

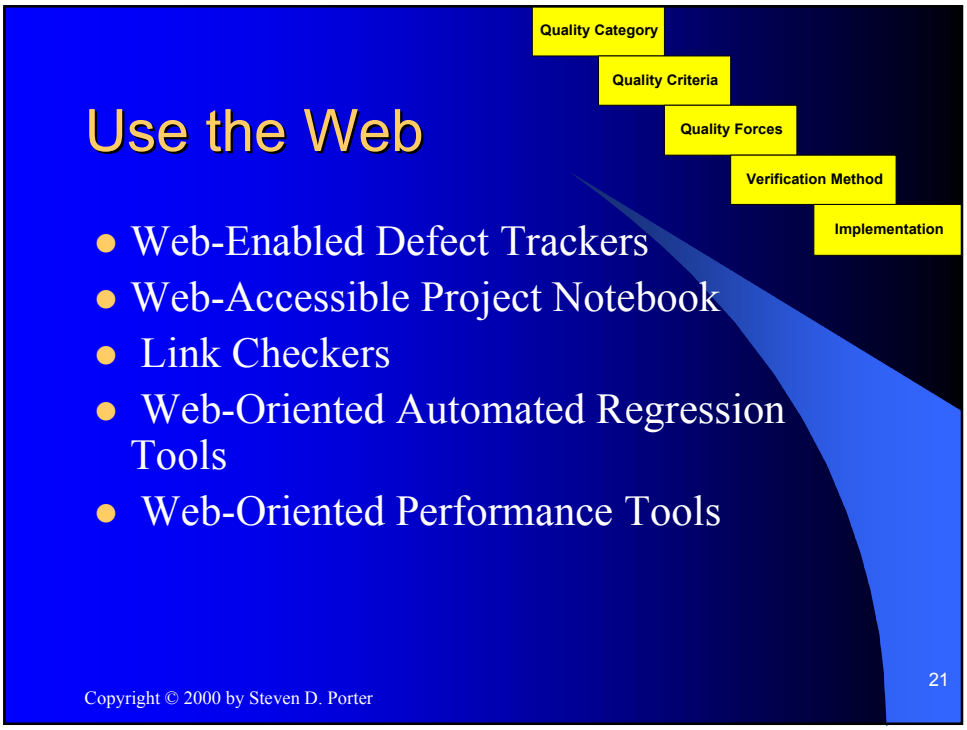

Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Manage Complexity
Competing Internal Business Interests
Development Environment
User Environment
Technology
Hardware Architecture
Software Architecture
Interactions between Quality Categories
Low
High
Copyright © 2000 by Steven D. Porter
16

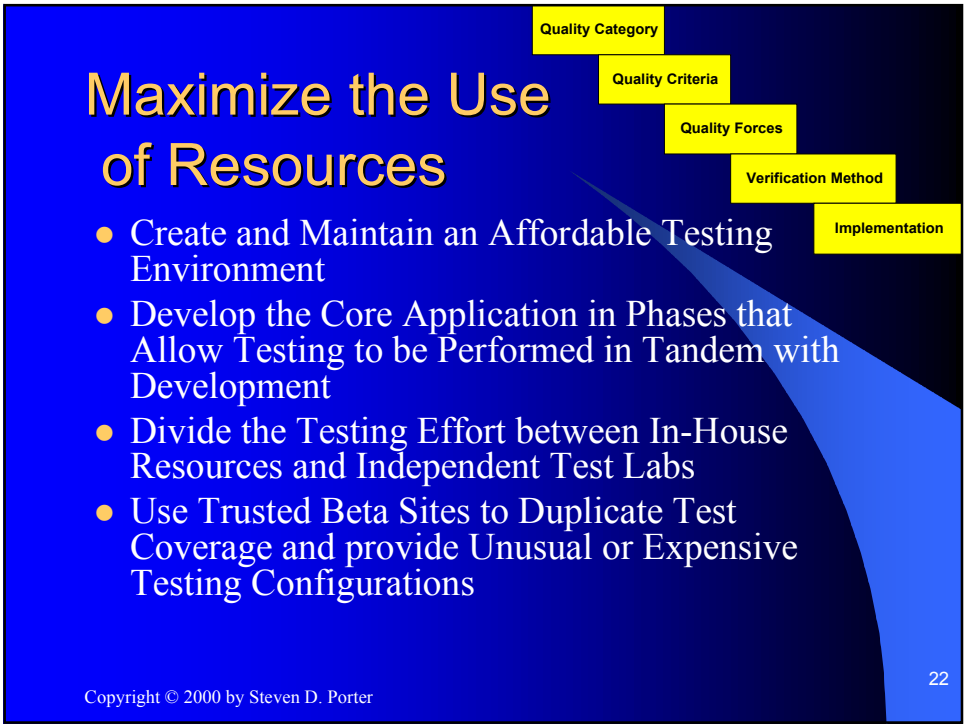

Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Reduce Risk
Affects
All Quality Categories
All Development Related Processes
Estimates and Schedules
Identify Analyze Mitigate
Less
More
Copyright © 2000 by Steven D. Porter
17


Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Negotiate Cost & Time
Increase as Volatility Increases
Increase as Complexity Increases
Increase as Risk Increases
Factor in the Impact of Compressed Schedules
Copyright © 2000 by Steven D. Porter
18


Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Strong Business Analysis
Promotes Understanding of the Business Objective
Used to Establish Standards, Success Criteria, and Verification Methods
Reduces Volatility
Allows for Many Solutions
Transcends Technological Changes
Maintains Consistency Over Time
Decreases Complexity
Communicates the Same Information to Different Development Teams or Different Members of the Same Team
Copyright © 2000 by Steven D. Porter
19

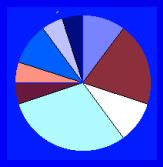

Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Determine Verification Method
Manual or Visual Review
Spell Checkers
Link Checkers
Site Monitors
Manual Testing
Automated Regression Testing
Performance Testing
Beta Testing
Copyright © 2000 by Steven D. Porter
20


Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Use the Web
Web-Enabled Defect Trackers
Web-Accessible Project Notebook
Link Checkers
Web-Oriented Automated Regression Tools
Web-Oriented Performance Tools
Copyright © 2000 by Steven D. Porter
21


Quality Category
Quality Criteria
Quality Forces
Verification Method
Implementation
Maximize the Use of Resources
Create and Maintain an Affordable Testing Environment
Develop the Core Application in Phases that Allow Testing to be Performed in Tandem with Development
Divide the Testing Effort between In-House Resources and Independent Test Labs
Use Trusted Beta Sites to Duplicate Test Coverage and provide Unusual or Expensive Testing Configurations
Copyright © 2000 by Steven D. Porter
22

The Strategy in Practice

- Theme
- Content
- Presentation
- Navigation
- Data Exchange
- Performance
- Technology
- Security



Theme

Standards	Corporate Image Site Objective
Volatility	Low Volatility
Complexity	Depends on objective (Inform vs Exchange)
Risk	Consistency with Overall Corporate & business objectives.
Verifications	Corporate Image Guidelines Standards etc. Traceability to site objectives
Implementation	In-House Manual Review

Content

Standards	Writing Style Language Usage Legal Constraints Accuracy, Completeness, Relevance
Volatility	Potentially volatile
Complexity	Variable - Depends on audience and message.
Risk	Moderate – Depends on rate of change
Verifications	Traceability to Objectives Spelling Syntax Grammar Logical organization etc.
Implementation	Manual Reviews Spelling & Grammar Checkers Professional Editing

25

Presentation

Standards	Graphic & Visual Standards, Corporate Image Guidelines, Fonts, Logos, Trademarks, Target Video Display Standards
Volatility	Moderate – Depends on supported browsers, platforms & versions.
Complexity	Moderate to High – Depend on Html usage (frames etc) Target browsers Dynamic page presentation The use of plug-ins
Risk	Moderate – Affects Performance, Consistency, Reliability etc.
Verifications	Verify that all features of the site /Application work in the supported browsers. (Compatibility Testing)
Implementation	Consider a strategy that supports in-house compatibility testing on current versions of major browsers, and out-source compatibility testing of all other browsers

26

Navigation

Standards	Site architecture Navigation standards User Interface guidelines
Volatility	Depends on site objectives and the outside links
Complexity	Low to High Sequential Hierarchical Grid Web-Linked
Risk	Low to Moderate – Outside links, unstable site architecture
Verifications	Manual review of architecture Verify functioning of all links
Implementation	Use link checkers to: Verify links Identify orphans Slow downloads

27

Data Exchange

Standards	Accuracy & Completeness, N-Tier Architecture, Coding & Validation
Volatility	Low to Moderate for the exchange mechanism.
Complexity	Moderate to High - Depending on N-Tier & Server Environment
Risk	High – due to complexity and web
Verifications	Verify that communication objects pass data correctly between client and the servers, as well as between servers Verify that Web Application handles Add, Modify, Delete and Query capabilities correctly
Implementation	Functional Regression testing Manual techniques Automated techniques

28

Performance

Standards	Established project requirements Industry standards User expectations
Volatility	Low – If changes comply with the standards for performance. Performance stability is affected by the rate of change of the hardware, software, site content and structure
Complexity	Moderate to high. Depends on the Architecture
Risk	High – Concurrent access to the site and database is higher than predicted. Hardware is insufficient to handle load. Application is not designed for optimum performance
Verifications	Volume, Load and Stress Testing across the Web.
Implementation	Use In-House owned performance tools that verify performance for 200 virtual users submitting distinct requests. Outsource performance tests for higher numbers of users

29

Technology

Standards	Minimum requirements for hardware and software configurations. Supported environments
Volatility	Moderate to High – We can manage the rate of change of the configurations we control. We have no control over the rate of change of technology, nor the use of it by our customers.
Complexity	High – Client/Server Hardware coupled with firewall implementations create a complex application environment.
Risk	High – The broad spectrum and rapid change of technical solutions causes the system environment to be unpredictable and uncontrolled
Verifications	Testing configurations. Review changes to server-side and evolving client configurations
Implementation	In-House compatibility testing for primary supported environments. Out-source compatibility testing of marginal environments. Perform beta testing at beta sites with different firewalls.

30

Security

Standards	Corporate Security Policies Fiduciary Responsibilities Purpose of Site
Volatility	High – Changes caused by new methods to bypass current technology
Complexity	High – Various encryption schemes Various levels of security access to users and groups of users
Risk	High –Breaches of security may be catastrophic to the business and the users
Verifications	Automated and manual monitoring
Implementation	24/7 monitoring Immediate notification for security breaches Mitigation Plan

Summary

- Web Sites are increasingly becoming complex Web Applications
- Establish a Clear Business Direction and Purpose
- Manage the Quality Forces
- Divide and Conquer


Contact Information
Steven D. Porter
3034 Estepa Dr.
Cameron Park, CA 95682
Sporter@iventex.com
Thank you to David Wilkerson for the hours spent helping me organize, clarify and edit this presentation.
Dwilkers@calweb.com
Copyright © 2000 by Steven D. Porter
33

From Website to Web App: Ensuring Quality in a Complex Environment

Introduction

Marshall McCluhan, a professor of communication from the University of Toronto, in trying to explain the effect of the mass media on our lives, coined the phrase, "The Medium is the Message". At the time his ideas were popular (1960s and 1970s), he theorized that television, radio and the print media help define the way we schedule and alter our lives as much as the information broadcasted. For example, people tend to make phone calls during tv commercial breaks, thus causing a spike in the amount of traffic over the phone system. From a social aspect, each medium affects the way we communicate with our family, friends, and associates on a daily basis. For example, I find myself tuning into the conversation on the radio as I' m traveling with my family rather than talking with the family. This is a function of the medium, and how I use it, rather than the actual information presented by the medium. Furthermore, with limited delay, we can actually experience much of the day-to-day realities of life from all parts of the globe at any time of the day.

The web is an extremely young volatile medium when compared to television, radio and print, but has already shown its impact on our lives. It has changed the way we communicate with our friends, perform business, and use our disposable time. Governments are regulating it, considering tax and political strategies around it, and determining the best methods of defense from it and through it. For good or for bad, the internet is literally invading every aspect in our lives. The words of Marshall McCluhan, "The Medium is the Message", have never been more relevant.

The business community has recognized the potential value of the web for increasing profits. The barriers of entry are still relatively low. Anyone with an idea, and the desire to promote it, can publish it on the web in a matter of minutes. From the consumer side, we can access information on virtually any subject and purchase anything from books to cars to real estate. New ways to use the web, to support the infrastructure, and to harness its power are developing at an unprecedented rate. In our headlong rush to maximize its potential, we are asking a relatively new, immature medium, to support communication and commerce in a very mature way. We expect the same level of excellence from the web as we have come to appreciate in the older technologies of television, radio and print.

The risk of depending on a rapidly changing, immature medium is similar to the risk we assume when we expect a child to act like an adult. Sometimes the results are spectacular, other times they are downright dismal. Some web oriented companies, initiatives, or forms of entertainment, are highly successful, while others are posting record losses. Some great websites go unnoticed, while others generate incredible excitement. A new Internet startup is flush with cash in one year, and the next year it is fending off bankruptcy. But we take the risk anyway knowing the odds of success are against us. The web is enticing, the rewards are high for those who succeed, and the initial effort and cost to startup on the web seems low in comparison to other more conservative, proven

choices. We get to experience the entrepreneurial rush of exploring uncharted territory. "To boldly go where no man has gone before".

We are not going to stop developing good ideas and businesses that depend on the web simply because it is not quite ready. The way to lower the risk of failure is to apply the mature concepts we have developed over centuries of communication and commerce to the new medium of the Internet.

Websites and Web Applications – The Same Thing Only Different

Conceptual View

There are two main methods of interaction defined in any verbal communication.

1. Monologue
2. Dialogue

In the first category, a person merely presents information, and receives feedback from the audience through nods, smiles or even thrown tomatoes. Political speeches, lectures and sermons fall into this category. In the second category of communication, a person initiates a conversation, expecting interaction with another person or group. Direct Sales and Roundtable discussions fall into this category.

The targeted use of every medium incorporates these two mechanisms. Television and radio tend to be monologues, although the use of rating services and rise of talk shows generates a level of feedback. The print medium tends to be the one-way presentation of ideas with purchases being the primary feedback mechanism. The exchange of letters is an example of written dialog. The electronic forms of print combined with letters have created the dialog we experience through email, chat rooms and the like. The combination of communication type and the degree of feedback involved constitutes the essential foundation of any medium of communication.

In terms of eCommerce, the two forms of communication are represented by:

1. Websites
2. Web Applications

In many respects, a web application is merely a website with added capabilities. Some development shops view them as one in the same, creating opportunities for quality gaps. For the purposes of clarity, we will distinguish between the two forms of communication by their purpose and function. Websites are for the purpose of presenting information, whereas Web Applications present information and promote dialogue or interaction between the customer and the business.

In a website, users open the home page, search for information, view graphics, hear audio files, and generally peruse the site for what interests them. They may be encouraged to call for further information or send an email. An example of this type of site is www.allhealth.com, a site that provides medical information. The Internet is deluged with these types of sites. Personal home pages are another example of websites.

Web applications, on the other hand, are the natural result of the effort to promote dialogue across the Internet. The primary purpose of this dialogue for business is to sell a product or service. A customer is presented with information and encouraged to order online. The customer chooses an item and selects a method of payment. Pertinent data for each transaction is passed back and forth between the customer and the business until the transaction is complete. An example of a web application is www.Amazon.com, an

online business that provides a catalogue of information about books, cds, etc., and a web interface for the purchase of those items.

As we move from the monologue approach of websites to the dialogue approach of web applications we significantly increase the amount of complexity to the relationship between the participants. Where once we were able to limit our effo rts to the publishing of a sophisticated online organization of linked ideas and information, now we expect and promote direct interaction and involvement from our current and potential customers. We collect information about our customers and their orders, storing that information so that we may build a lasting relationship in which the only interface between the business and the customer may be the web application itself. If the web application is well designed for this purpose, then the relationship is a prosperous one.

Technical View

Websites and web applications are different from a technical view as well as a conceptual one. Websites are collections of static pages built with html and associated through hyperlinks. The site is accessed through web browsers using an http protocol. Many sites provide secured access.

Web applications contain the same basic elements of web sites, however they are also very similar to client/server applications. Web applications allow for user input and standard database transactions (read/write). Business logic is performed as well as client-side validation of data. Queries can be initiated, as well as dynamically generated reports displayed. The security for these sites tends to be implemented in a more complex and sophisticated manner than for websites.

The web is a means to achieve platform independence. We are not required to have a specific type of operating system to access the web and it's incredible amount of information. We only need a web browser on the client and a portal to the web to access the data on a site server.

As the sophistication of the applications increase, the constraints of the current web technology have more impact. We are hampered by the lack of standardization within the browsers themselves rather than the platforms. Competing web browsers do not uniformly display content. They do not perform logic the same way, and they do not accept user input in quite the same manner. For example, some browsers do not support Java scripts. Objects such as buttons and grid controls react in different ways. File downloading is handled through alternative strategies. This lack of standardization makes it difficult for us to confidently predict the appearance of our website or application to our customers, and whether it will perform correctly. Furthermore, we sometimes require plug-ins such as Shockwave or QuickTime to be loaded on the client machine to achieve maximum impact. This dependence on 3rd party software increases the risk that our customers, with unknown technical knowledge and hardware configurations, may be incapable of using our site effectively. In this case our web site or application will be of limited help to our business and customers.

The Quality Strategy

Our challenge as Quality Assurance and Quality Control professionals is to help create successful websites and web applications that fulfill their mission in spite of the **V**olatility of the web, the **C**omplexity of the implementations, and the associated **R**isks of doing business on the web. Any strategy we use must take into account these **VCR** forces (volatility, complexity and risk) and minimize their impact on our effort. Recognize that the web is volatile, and that the transition from websites to web applications increases the complexity. Projects must deal with not only complex technology but also security and privacy in a way that will minimize the overall risk. Finally, be aware that market demands compress development schedules increasing the likelihood of error.

The strategy for managing a quality control effort in a web environment involves applying best practices to the web development effort.

1. Organize the quality effort into categories
2. Establish the standards of quality for each category
3. Determine the verification methods.
4. Establish a review process
5. Manage the VCR forces (volatility, complexity and risk)
6. Negotiate cost and time
7. Implement the strategy
8. Manage and leverage all resources to accomplish the task

Divide and Conquer

The first step toward outlining a specific strategy for quality is to divide the effort into manageable pieces much like dividing a pie into edible portions. The categories chosen depend upon the purpose or mission of the web site. The type, priority, criticality and importance of each category will influence the amount of effort required to achieve high quality. The VCR forces will influence our ability to achieve a desired level of assurance. Cost and time will influence the amount of effort we can afford and how that effort will be administered within each category. Some examples of verification categories are:

Theme	Data Exchange
Content	Performance
Presentation	Technology
Navigation	Security

Theme - The core message and objectives of the site in the context of the business.

Content – The visual, auditory and written information that realizes the site theme.

Presentation – The look and feel of the content and the site. Includes such things as visual appeal, content flow, and internationalization.

Navigation – The means by which a user traverses the site.

Data Exchange – The means by which discrete pieces of information is passed to or collected from users, validated and retained by the site.

Performance – The speed at which a site can be navigated under varying conditions.

Technology – The underlying hardware and software that is necessary to meet the site' s objectives in delivering the content, presentation and exchange of data.

Security – The methods used to assure integrity and reliability of data, control access, maintain confidentiality of information and assure the continued reliable operation of the site.

Create Standards and Evaluation Criteria

The second step is to create quality standards for each category. The standards reflect the need to satisfy user and business expectations, and are tempered by technical constraints. Quality to the user may include requirements for:

Responsiveness	Relevance
Accuracy	Aesthetics
Completeness	Security
Ease of Use	Privacy
Reliability	

Any standards must incorporate the expectations of the business such as:

1. Customer Impact
2. Sales Generation
3. Business Credibility
4. Cost
5. Time to Delivery
6. Reliability and maintenance

For example, the standards for web application performance must encompass the discrete sales generation desires of the business (e.g. 1000 completed financial transactions per hour), and the polled or predicted expectations of the user community (e.g. less than 3 seconds to load static pages).

Implement a Review Schedule

Develop standards and their evaluation criteria for the development process as well as for the application. Candidates for review are:

1. **P**rocess
2. **A**rtifacts
3. **R**oles
4. **T**ools
5. **S**tandards

Two examples of processes are requirements elicitation and change management. Each activity of the processes has input and output artifacts. Individuals or groups perform various roles. Tools are used to accomplish the tasks. We review and evaluate all of these parts to verify they are performing as planned, as well as managed to the needs of the fast paced web development lifecycle. Constantly evaluate whether the standards you have set are correct and attainable.

Manage the Impact of the VCR Forces

Though there are many similarities between web applications and client/server applications, there is at lest one main difference. This is the underlying assumption that the websites and web applications are built to incorporate change whereas client/server applications remain as unchanged as possible. For example, content to a website may change daily or more often. Web server technology may change every couple of months as load is increased. The number of users may increase dramatically. Furthermore, due to high demand for services, developers and webmasters come and go as quickly as higher salaries are offered from the competition. In comparison, client/server applications appear absolutely frozen. The content changes little, the system is set up to manage a predictable stable population, and change to the technology is discouraged due to the amount of cost and work to update the client/ server infrastructure corporate-wide.

The built-in volatility of web applications represents a significant challenge for the quality assurance / control professional. One way to handle this challenge is to separate the volatile elements of the project and product from the non-volatile ones. For example, we don' t expect the theme to change much from month to month. However we expect the content to change often. We therefore need to setup a process of review that handles the highly volatile content more aggressively than the more stable theme.

Some examples of non-volatile elements are:

1. Business Model
2. Application' s Architecture
3. Theme

Some examples of volatile elements are:

1. Content

2. Technology
3. Personnel

Complexity creeps into the project in a myriad of ways. Just determining the theme can be difficult due to internal business interests with differing visions and goals, such as marketing, with its long-range efforts, and sales, with its short-term needs. Other areas of complexity include the:

- Development Environment
- User Environment
- Technology
- Hardware Architecture
- Software Architecture
- Dependencies between Quality Categories
- Internationalization
- Internal political agendas

Risk affects all the quality categories. It affects the development related process and the estimates and schedules. On the flip side, the choice of the development processes as well as the estimates themselves can alter risk. Reduce risk by identifying the sources of risk, analyzing their impact on the project, and providing a plan to reduce the affect of a realized risk. For example, one way to reduce the damage created by a server crash is to provide some form of redundant server configuration.

There is never enough money or time to do the job as well as we would like. Even NASA has its budgetary constraints. In order to get the maximum advantage, negotiate cost and time with the business decision makers. In general increase the amount of cost and time as:

- Volatility increases
- Complexity increases
- Risk increases

Factor in the impact of compressed schedules demanded by the market. Consider alternative ways to use increased budget to offset limited time or resource availability by outsourcing a portion of your work to professional testing labs.

Promote Business Analysis - Its Good Business

With every good idea, there is a plan to implement it. In the software industry we are sometimes guilty of taking a problem or idea, and designing a solution or application without fully understanding the vision. We do this in order to "speed delivery". Some of the modern methodologies such as rapid application development seem to promote the idea that we can begin coding with limited knowledge. Some groups implement RAD in such a way that business analysis is limited, design is forgotten, or testing is not considered.

Imagine trying to build a bridge without understanding its purpose and predicted load conditions. The subsequent disaster would be reported in the front page of every newspaper complete with body counts. But some of us look for ways to bypass important steps in the software lifecycle. We increase the risk of failure with every bypass of essential steps. Websites and web applications need the same diligence paid to the important activities of business analysis, design and testing as other mission critical projects.

The business model may be the most reusable work of the project. The business model may be the least volatile of all artifacts during the life of the software. The model transcends technological changes, allows for many solutions, and maintains consistency over time. Strong business analysis communicates the same information to different development teams, evolving teams, or different members of the same team, thus reducing the negative effect of the VCR forces.

Once we have an adequate business model we can validate the site's objectives and theme, obtain customer buy-in, establish success criteria and choose verification methods. Maintaining our focus on the business objective will help us build the right site.

Verification Methods – The Means to an End

There are several types of verification methods from which to choose:

Manual or Visual Review	Manual Tests
Spell Checkers	Automated Regression Tests
Link Checkers	Performance Tests
Site Monitors	Beta Tests

We may include the use of one or more than one of the methods identified to verify the quality for each category. For example, content may require manual review, spell checking, and link checking. Security may require manual tests, beta tests and automated regression tests. Theme may only require a manual or visual test.

When considering the implementation of the strategy and the verification methods, use the capabilities and tools associated with the web. The use of web-enabled defect

trackers, web accessible project notebooks, link checkers and other web oriented automated tools will aid immensely in supporting the quality strategy.

Maximize the use of resources by:

1. Creating and maintaining an affordable testing environment
2. Developing the core application in phases that allow tests to be performed in tandem with development of code
3. Dividing the test effort between in-house resources and independent test labs.
4. Using trusted beta sites to duplicate test coverage and provide unusual or expensive testing configurations.

The Strategy in Practice

Suppose an investment firm wants to create an informational website. The marketing department is interested in a website where current customers can view relevant information about the investment firm, access a weekly newsletter (which is already sent by mail), and navigate to pages about each investment opportunity highlighted in the newsletter. An archive of all newsletters will be maintained for research and investment purposes. The sales department would like potential customers (or guests) to also visit the site. Sample newsletters from archived material will be available to the guests. Only paying customers will have access to the current newsletter. Guests will have the ability to pay a subscription fee online. Current customers include citizens of many countries. Thus the site must be multi-lingual and a solution for financial transactions must support multi-currency.

The board of directors, though they are committed to the initiative, would like to know whether the investment in the website is increasing business, or diverting valuable resources from other proven marketing and sales tactics. The board is concerned because they are unfamiliar with the costs of creating and maintaining a website. If, over two years, the website proves profitable, they will continue supporting the effort. We have been given 6 months to deliver the website. At this point, cost has not been estimated, though some money has been placed in an account to use as startup capital.

This website is actually a web application. Though the requests appear relatively simple, a number of technical challenges await the developers. For instance, the newsletter is changing weekly as well the links to detailed information about each investment opportunity. Guests may view sample material, but there must be some way to distinguish them from paying customers; i.e. some form of secured access. Guests may pay for the service online – there is a requirement for secured transactions. A database of archived material points to a need to manage data storage. We can only estimate the possible number of concurrent hits to the site. Some method of correlating site statistics to sales needs to be implemented. Last, but not least is the request for an international site that handles multi-currency transactions.

Since the investment firm is unfamiliar with costs and the effort required to develop this project, there is a potential of insufficient funds. Furthermore, the deployment date has been dictated rather than estimated. This might prove injurious since we don't know the full scope of the project, we have not chosen a set of development tools, the hardware and software configuration of the web application has not been designed, and the project team has not been assembled.

The complexity of the web application, the rapid change of content, and the risks introduced by the board of directors contribute to the factors working against success of this project. But we move forward anyway, knowing that high-risk projects also produce great rewards when successful.

Even though we have a vision of the site, we have not pinned down all the requirements and business rules. Nor have we determined how the board of directors will determine whether the project is a success. These tasks can be done while we are preparing our test plan and environment.

As the test manager on this project, begin by validating the business objectives and success criteria with the project manager and/or key stakeholders. Procure written confirmation of these items. Then plan the test effort by dividing the project into essential categories for evaluation.

Categories

Theme	Data Exchange
Content	Performance
Presentation	Technology
Navigation	Security

Set standards for evaluation in each area. To avoid meaningless tests, verify that the methods and standards of evaluation for each category support the site and business objectives as well as the success criteria.

Estimate the amount of volatility and complexity for each area, and identify the associated risks. Plan the type of verifications that will be used and how they will be implemented. The following tables show the results of this planning for the web example. They are not intended to be complete, but rather they are intended to present a method of organizing the high level information.

Theme – The core message and objectives of the site in the context of the business.

Investment Newsletter

Standards	Business Objective
	Site Objective
	Corporate Image
Volatility	Low
Complexity	Low
Risk	Maintain corporate and business objectives
Verifications	Corporate Image Guidelines, Standards etc
Implementation	In-house manual review

Content – The visual, auditory and written information that realizes the site theme.

Changes Weekly

Standards	Writing Style
	Language Usage
	Legal Constraints
	Accuracy, Completeness, Relevance
Volatility	High – Weekly changes to the content
Complexity	Moderate – market is sophisticated investors
Risk	Low – company has proven quality in newsletter publication
Verifications	Traceability to objectives
	Spelling
	Syntax
	Grammar
	Logical organization
Implementation	Manual Reviews
	Spelling and grammar checkers
	Professional editing

Presentation - The look and feel of the content and the site. Includes such things as visual appeal, content flow, and internationalization.

Standards	Graphic and visual standards
	Corporate image guidelines
	Fonts
	Logos and Trademarks
	Target Video Display
	Internationalization
Volatility	Low – More investigation is needed to determine supported browsers, platforms and versions of these elements
Complexity	Moderate – More investigation into the supported technologies (browsers, platforms) and features such as:
	HTML Usage (Frames etc.)
	Dynamic page presentation
	Use of plug-ins
Risk	Moderate – Affects performance, consistency and reliability

Verifications	Verify that all features of the web application work correctly in the supported browsers (compatibility testing)
Implementation	Consider a strategy that supports in-house compatibility testing on current versi9ons of major browsers, and outsource testing on all other browsers. Create a plan that supports testing the application on future browser offerings as they become available.

Navigation - The means by which a user traverses the site.

Standards	Site architecture Navigation standards User Interface guidelines
Volatility	Low to Moderate – Basic links can be stable, however, external links may change as well as links within the changing newsletter
Complexity	Unknown at this time. The team has not determined the type of site navigation to be implemented: Sequential Hierarchical Grid Web-linked
Risk	Low to Moderate – Outside links or an unstable site architecture can create errors
Verifications	Verify chosen site architecture meets needs of users Verify site meets the site architecture (map) requirements. Check for broken links, orphans etc.
Implementation	Visual review of site architecture Use automated link checkers to find broken links, identify orphans and slow downloads. Manually verify links that are missed by the automated site checkers.

Data Exchange - The means by which discrete pieces of information is passed to or collected from users, validated and retained by the site.

Standards	Accuracy & Completeness N-Tier Architecture Coding & Validation
Volatility	Moderate – depending on the technology used to enable the exchange.
Complexity	Moderate to high – the transactions being processed are relatively commonplace now –security and personal information is being exchanged. There are no calculations other than subscription dates. Though the technology is evolving, there are good and stable choices to handle these elements of the application.
Risk	High – due to the complexity and volatility of the web itself.
Verifications	Verify that communication objects pass data correctly between client and the servers, as well as between servers
Implementation	Invoke manual and automated methods for: Functional testing Regression testing

Performance - The speed at which a site can be navigated under varying conditions.

Standards	Established project requirements Estimates of usage for each functional area of the web application Industry standards Polled user expectations
Volatility	Low – Changes to the application and server environment must comply with the standards for performance.
Complexity	May be high since some standards have not been determined as well as the deployment plan
Risk	High – Concurrent access to the site and database may be higher than predicted. Hardware may be insufficient to handle the load Application must be designed for optimum performance within a complex web environment
Verifications	Verify performance against the standards and expectations.

	Verify hardware and software configuration is designed and implemented for optimum performance
Implementation	Implement automated tools for volume, load and stress testing across the web. Use in-house owned performance tools to verify performance for 200 virtual users submitting distinct requests at the same time. Outsource performance tests for larger numbers of users.

Technology - The underlying hardware and software that is necessary to meet the site's objectives in delivering the content, presentation and exchange of data.

Standards	Minimum standards for hardware and software configurations. Supported client configurations
Volatility	Moderate to high (depends on our target customers). We can manage the rate of change for the configurations we control. We have no control over the rate of change for the industry, the web, or the use of it by our customers.
Complexity	High – Client/Server Hardware coupled with firewall implementations and multiple platform/browser configurations
Risk	High – The broad spectrum and rapid change of technical solutions causses the system environment to be unpredictable and uncontrolled
Verifications	Prove the deployment design Constantly monitor and review changes to server-side and evolving client configurations
Implementation	In-house compatibility testing for primary platform/browser configurations. Perform proof of concept of the deployment design including hardware, and off the shelf software used in the deployment Outsource testing for marginal environments Perform beta tests at corporate beta sites that implement differing firewalls

Security - The methods used to assure integrity and reliability of data, control access, maintain confidentiality of information and assure the continued reliable operation of the site.

Standards	Corporate security policies Fiduciary responsibilities
Volatility	High – Changes to the security implementation are caused by the need to prevent breaches created when hackers bypass current technology
Complexity	High – Two main areas of concern: Security that allows only authorized users into specific areas of the site. Security that protects the privacy of individuals who purchase their subscriptions through the web application
Risk	High – Breaches of security may be catastrophic to the business and users. Potential of lawsuits
Verifications	Prove security model prior to deployment Automated and manual monitoring of security
Implementation	Functional testing of security Consider hiring a firm that specializes in web security to create and monitor this aspect of the site 24/7 monitoring Immediate notification to IT staff and corporate officers Mitigation plan

Summary

The demands of the marketplace and the evolution of technology are changing the way we use the web. Informational websites are now becoming highly interactive web applications. Our job as QA and QC professionals is becoming more difficult as we try to grapple with the forces of volatility, complexity and risk that affect the development effort and success of the application. We can use known methods of organization to help us manage the testing effort. First establish a clear business direction and purpose for the application. Divide the effort into areas of concern. Establish standards and success criteria for each area. Calculate the impact of the VCR forces to the project and manage your resources accordingly. Determine the methods of verification that will be used on the project and create an implementation strategy. As much as possible, use automated

technology designed for the web. Consider the use of outsourcing the tests that are complex, marginal or otherwise difficult and expensive to achieve within the internal testing environment. Use beta sites for testing the web application on divergent corporate network configurations. Establish a method of ongoing review that evaluates the essential process, artifacts, roles, tools and standards for your project.

QWE2000 Session 12M

Ms. Tuija Lamsa [Finland]
(University of Oulu)

"Using Knowledge Management in the Quality Improvement of the Development Processes"

Key Points

- Utilization of Knowledge management in software business
- Individual and organizational intelligence and learning
- Quality improvement in the context of development processes

Presentation Abstract

Today's business environments are more and more complex of their nature. Organizations are undergoing continuous changes and needs in their markets, information technology, interest groups and competitors, and in the management of processes and intangible assets. These essential things force the companies in different industries to change their ways of actions and processes, and to focus increasingly on both quality and performance improvement and the utilization of multidimensional intelligence and intellectual capital both at individual and organizational levels.

Especially knowledge management (KM) has gained plenty of attention among practitioners and academies, and this topic has been approached from different points of view in different studies. KM can be considered as an interdisciplinary research, the domain of how we renew, success, improves competitiveness and harness the human talent and know-how for the use of organizations. Whether the notion of KM is defined in terms of learning, intellectual capital, knowledge assets, intelligence, or otherwise, managing all kind of knowledge is one of the most significant challenges in management. Managing knowledge, and using it for improving organization's development processes, can be seen as one of the most valuable way to increase organization's success.

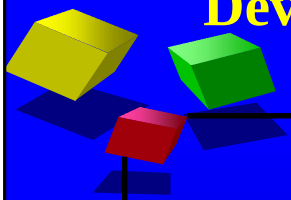
This paper focuses on using knowledge management in the quality improvement of the processes in the organizations, which presents a specific research challenge, for the reason that these business environments face the most intensive and highest rate of change. The purpose of this paper is to discuss how knowledge management can be utilized in the context of innovative and turbulent conditions, in the R&D development processes. The paper aims to increase understanding of the implementation of knowledge management and its importance in development processes. The paper also pursues to develop a system of concepts for describing

and analysing the management of knowledge in the above-mentioned context, in the software business.

About the Speaker

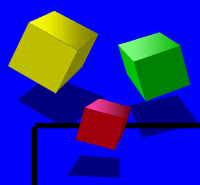
The author's field of research is in Knowledge Management, Individual and Organizational Know-how/Intelligence and Quality focused on software and R&D business environments. Tuija Lamsa is a M.Sc., economist, who has completed her Masters Degree in Economics and Business Administration, as her subjects being Marketing, Business Law, Accounting and Industrial Management. Her current post is in University of Oulu, at Faculty of Economics and Industrial Management, Department of Management and Entrepreneurship where she works as a Researcher and Assistant of Quality Management. She has lectured in Organizations and Management, and Quality, and currently she is working in co-operation in projects dealing with Knowledge management and Process improvement.

Using Knowledge Management in Improving the Quality of the Development Processes



Tuija Lämsä
The University of Oulu
Finland

4th International Software Quality Week Europe (QWE2000)
Brussels, Belgium
20-24 November 2000



The Focus of Research

- **The utilization of Knowledge Management in improving the quality**
- **The role of management in the context of innovative and turbulent conditions, in the R&D environment**
- **Managing organization's knowledge assets**

Definitions of Knowledge Management

Leonard-Barton	Knowledge Management allows you to determine the explicit knowledge that is somewhere in your organization that you can find and leverage rather than having to reinvent the wheel.
Novins	Organizing information from disparate sources into a context that reflects the business and the decisions and processes of the business.
Applications, markets and technologies	The task of developing and exploiting an organization's tangible and intangible knowledge resources. Knowledge management covers organisational and technological issues.

Challenges in Quality Improvement & New Revolution of Knowledge

"Writing software is an art, not a science, and must be managed as a craft, if it can be managed at all"

By Blackburn, Scudder and Van Wassenhove, 1995

- The nature of business environment
- The nature of software
  - amount, complexity, criticality
- Pressures of developing systems (>> demands for higher quality and performance)
- Globalization
  - global market, emerging markets in developing countries, importance of Internet and R&D
- Consolidation

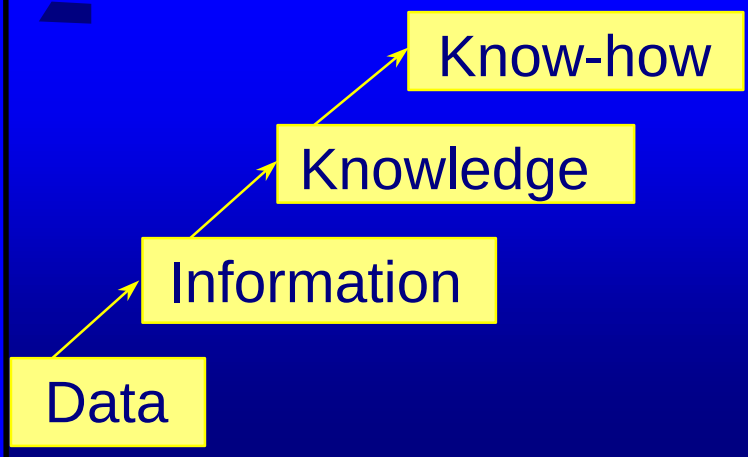
Changing Role of the Management

”Management has two sets of problems; those of today and those of tomorrow”

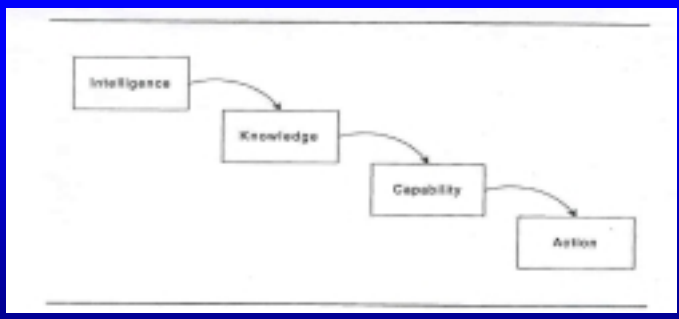
By Dr. Deming

- **New paradigm of management:**
  1) The change in factors of production
  2) Focus on knowledge assets and individual competences
  3) The dilemma of Knowledge Management

The Nature and Value of Knowledge



The Nature and Value of Knowledge - Intelligence as Generator for Selective Action



Tuomi, 1999

The Nature and Value of Knowledge – Tacit vs. Explicit Knowledge

- **Tacit knowledge:**
  - Context-specific, personal, hard to formalize and communicate, inherent in our very thinking, embedded in work >> unconscious value

- **Explicit knowledge:**
  - Written knowledge, easier to identify, re-usable in consistent and repeatable manner, can be stored as artefact (physical or virtual) in systems >> can be identified, measured, distributed and audited

The Nature and Value of Knowledge

"In an economy where the only certainty is uncertainty, the one sure source of lasting competitive advantage is knowledge."

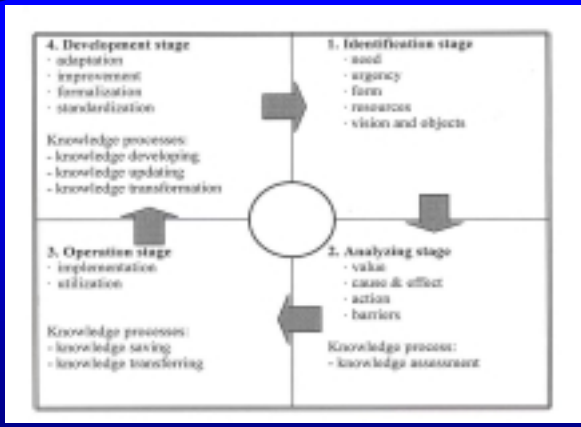
By Ikujiro Nonaka, 1991

- **The value and quality of knowledge**
  - **knowledge of new technology**
  - **knowledge of information processes**
  - **individual and organizational competences**
  - **knowledge has no intrinsic value**
  - **depends on a complex social system of activity, creating value using environment**
  - **difficult to estimate**

Managing Knowledge

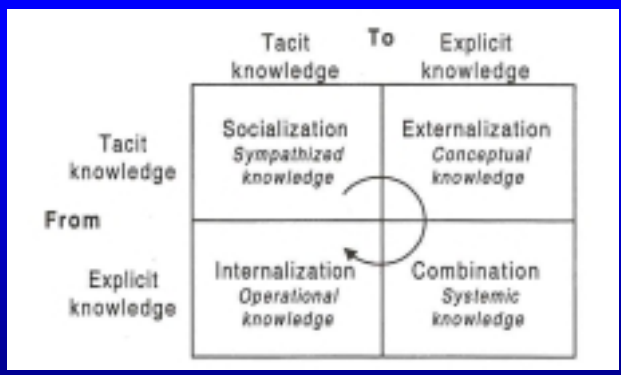
- **A presupposition for successful implementation of improvement approaches is the use of appropriate knowledge**
- **The need for the development of high-class competences is extensive >> strategic knowledge assets, knowledge *contents*: *know-how, know-why,* and *know-what* - are used in different strategic *contexts***
- **KM and TQM**
- **Process improvement techniques >> not a universal list for development of processes**

Managing Knowledge – Knowledge Management Process



Knowledge Management Disk

Managing Knowledge – Knowledge Management Process



Nonaka & Takeuchi, 1995

Using Knowledge Management in Improving the Quality of the Development Processes

Tuija Lämsä
Assistant of Quality Management
University of Oulu, Finland

Abstract

Today's business environments are more and more complex of their nature. Organizations are undergoing continuous changes and needs in their markets, information technology, interest groups and competitors, and in the management of processes and intangible assets. These essential things force the companies in different industries to change their ways of actions and processes, and to focus increasingly on both quality and performance improvement and the utilization of multidimensional intelligence and intellectual capital both on individual and organizational levels.

Especially knowledge management (KM) has gained plenty of attention among practitioners and academies, and this topic has been approached from different points of view in different studies. KM can be considered as an interdisciplinary research, the domain of how we renew, success, improves competitiveness and "harness" the human talent and know-how for the use of organizations. Whether the notion of KM is defined in terms of learning, intellectual capital, knowledge assets, intelligence, or otherwise, managing all kind of knowledge is one of the most significant challenges in management. Managing knowledge, and using it for improving organization's development processes, can be seen as one of the most valuable way to increase organization's success.

This paper focuses on using knowledge management in the quality improvement of the processes in the organizations, which presents a specific research challenge, for the reason that these business environments face the most intensive and highest rate of change. The purpose of this paper is to discuss how knowledge management can be utilized and what is the role of management in the context of innovative and turbulent conditions, in the R&D environment, in software business. The paper aims to increase understanding of the implementation of knowledge management and its importance through development processes. The paper also pursues to present the main issues to be considering in managing knowledge assets – both individual and organizational capital.

1 Introduction and Theoretical Background

Organizations are facing continuous changes and uncertainty in their business environments. They are realizing that if they want to succeed in today's business, it is important to maximize the use and utilization of the knowledge in the best possibly way. Companies embodies enormous extent of resources, not just concrete and "explicit

hardware", but also resources, which could call, for example, *knowledge assets*[1], intangible or tacit resources. One of the main matters for managing *knowledge resources*[2] is diffusion of knowledge within organizations. These knowledge resources reside in many different places such as: databases, knowledge bases, processes, and above all, in people' heads. Organizations need to know what their knowledge resources are, and how to manage these.

During the last few years, there has been a lot of hype around knowledge management (KM). Like Peter Drucker has said: *"The effective management of knowledge is the key management challenge of the late 20th century"*, and I think that KM can be seen as a critical tool also for the 21st -century organization. KM is increasingly gaining respect among companies as a tool to increase profits, reduce costs, improve competitiveness, and develop additional markets. Whether it is defined in terms of learning, intellectual capital, knowledge assets, intelligence, know-how, insight or even wisdom, *"the conclusion is the same: manage it better or perish"* (Amidon, 1996). Knowledge management can be seen shifting from the logistics of storing information to the unexplored potential of the human imagination. There has been said, that KM is truly leading to innovation and the renewal of organizations.

There occur a great number of definitions to describe knowledge management in literature. One body of literature on KM has its origins in approaches to information technology, information systems and related issues. I have gathered few fundamental definitions about KM to describe its nature and meaning for development of the different processes within organization (table 1).

Table 1. Examples of definitions of Knowledge Management.

Author	**Definition of Knowledge Management**
Leonard-Barton, D. (1995)	*Knowledge management allows you to determine the explicit knowledge that is somewhere in your organization that you can find and leverage rather than having to reinvent the wheel.*
Novins	*Organizing information from disparate sources into a context that reflects the business and the decisions and processes of the business.*
Applications, markets and technologies (1998)	*The task of developing and exploiting an organisation's tangible and intangible knowledge resources. Knowledge management covers organisational and technological issues.*

According to above-mentioned definitions, it is crucial to specify how organizations manage the relevant knowledge, and how these knowledge assets are intertwined and utilized in the different processes of the business. Organizations are able to make theirs workers to operate more productive and break down the traditional organizational hierarchy

[1] Knowledge assets are the knowledge regarding markets, products, technologies and organisations, that a business owns or needs to own and which enable its business processes to generate profits (Macintosh, 2000).
[2] Different levels of knowledge, regarding the possibility to codify, can be recognized in organizations knowledge resources (Gore and Gore, 1999).

on its plant floors by using knowledge management. It is also, along with a number of other companies, harnessing KM to organize and understand the potential uses of internally developed technologies, with an eye toward marketing them to other companies.

On the operating level, individual is more likely to use the notion of KM as an umbrella term to refer to a host of technologies and decision-support tools. These include search engines, data mining, and expert systems. These technologies make it easier for corporations to seek out, sort through, organize, refine, disperse and share information. Other vendors provide software that supports these technologies. Knowledge management covers then organizational and technological issues.

For all the hype about these offerings, however, KM is more a way of doing business than it is a piece of software; it's more about business processes than systems. The technical solutions being offered under the KM rubric simply make it possible to address certain business issues more effectively and more comprehensively. According to several software managers, they claim that *"writing software is an art, not a science, and must be managed as a craft, if it can be managed at all"* (Blackburn, Scudder and Van Wassenhove, 1995).

There can be found pressures and challenges to improve quality in organization's development processes in today's turbulent business environments, particularly, in software industry. As contemporary business environment is becoming more and more complex, organizations face continuous pressure to change theirs functions and procedures. In addition to the ubiquitous nature of software, the amount of software code in most consumer products and systems is doubling every two to three years (Dutta, Kulandaiswamy and Van Wassenhove, 1996). Further, software developers are scrambling to cope with the pressures of developing systems which are not only a couple of orders of magnitude bigger and more complex than those developed a few years ago, but which also need to meet ever-increasing demands for higher quality and superior performance. Also the complexity and criticality of software within industry is high and continues to grow significantly every year, as software becomes an increasingly important component in many products (Maxwell, Van Wassenhove and Dutta, 1996).

The New Revolution of Knowledge and Information Technology

Globally, organizations appear to be remarkably similar in the way they structure the software process. Today, especially the information technology sector in Europe is growing up. The competition among these IT-organizations can be very rough and it is developing fast. There can be identified two trends, which characterize the world software industry – globalization and consolidation. The first means that location is becoming far less important than size for both developing and selling software. Consistently, consolidation means that mergers and acquisitions are increasing the market share of the largest companies – it has been estimated that some 90% of the global software market is now accounted for around ten companies, including IBM, Microsoft and Oracle.

According Gannon (1997), the information technology industry is a prime example of such a globalized industry; the cost of development, the cost of marketing and the powerful trend towards standardization combine to make the industry globally interdependent. Boutellier,

Gassman and von Zedtwitz (1999) have pointed that there is four most important drivers for globalization:

1. The global market;
2. Emerging markets in developing countries;
3. Emerging suppliers / work forces in developing countries;
4. Internet-based global communication driving the world to a 24-hour / 365day open market and bringing customers to businesses they would never see otherwise.

Especially, the importance of internet for globalization is unquestionable. As the information revolution continues, new possibilities for storing, retrieving and communicating information are created. World-wide access to information is facilitated by the Internet, new information technologies break down the walls both inside the company and to the outside world.

Traditionally among the most centralized functions of the firm, R&D is adjusting to world-wide dispersion of knowledge and technology creation (Boutellier et. al. 1999). Particularly, knowledge acquisition and know-how are of central importance in R&D. During the last century the amount of knowledge has tremendously increased, mostly through new and sophisticated information technologies. R&D can be considered as the most important element in industrial technology intensive organizations to source, store, create, transfer and diffuse knowledge. The organization of R&D must therefore be desingned to selectively retain information, process knowledge, and apply know-how.

First and foremost, I think that when organization decide to use knowledge management in their development processes, it must be adapted to organization's needs, not that image to 'must have' certain kind of new method to solve all problems. First of all, organizations must consider their own goals and objects, where they want to be in markets. Hence, I'm presenting some crucial elements and factors of knowledge management, which improve organization's capability to operate in best possible way through its developed processes.

2 Research methodology

Developing processes in software business have been approached from quite different perspectives; for example, being emphasized both, constructive or more reacting viewpoints. Development processes have been studied very often from the social constructivist view of the technology in literature. Researchers (e.g., Barley 1986; Jones 1990; Weick 1990) working with this point of view claim that the information technology is equivocal, i.e., that it can take on different forms and holds an infinite number of possible applications, and thus, needs ongoing sense making to be managed. And yet, a few of them have also been taken on a reactive approach to the information technology, as they analyze how organizations and people in them interpret and react to new IT-applications introduced to them. Furthermore, Vendelø (1998) has been adopted a proactive approach in his study, as it examines how a software company produces interpretations of its software platforms.

In this paper, I have adopted the institutional perspective, which brings quite a diverging, but still refreshing and new point of view in the discussion of developing and improving the

quality of the processes in R&D environment within IT-organizations. In recent years institutional theories and these application have been one of the growing field of interest in business economics, and particularly in organizational study (e.g., Scott 1995; Greenwood and Hinings 1996; Hinings, Greenwood, Brown and Cooper 1996). Earlier institutional study has been focusing mainly on public organizations, but the dimension of institutional research and theory is a great deal of expansive for the time being. Hence, present study is opening a challenging perspective to study knowledge management in such organizations who are operating in turbulent, unstable and high-risk circumstances.

Organizations are becoming more and more homogenous, for example, by their nature, structure and technology in their business environments. It can be assumed that almost every organizations in software business obtain the alike or of the same kind of technology. So, if these environments are fairly the same and the organizations in these organizational fields constitute a recognized area of institutional life: key suppliers, resource and product consumers, and other organizations that produce similar services or products, there is needed means and systems to make a difference in competitivity between organizations. Like DiMaggio and Powell (1983) have stressed nearly 20 years ago,

> Organizations may change their goals or develops new practices, and new organizations enter the field. But, in the long run; organizational actors making rational decisions construct around themselves an environment that constrains their ability to change further in later years. Early adopters of organizational innovations are commonly driven by a desire to improve performance.

But new practices are becoming continuously. Organizational changes in formal structure, organizational culture, and goals or processes are happen. Organizational change varies in its responsiveness to technical conditions (DiMaggio and Powell, 1983). In this paper I'm focusing in these development processes that improve quality, both in different processes and in organization in its entirety. Research and development is an institutionalized category of organizational activity, which has meaning and value in many sectors of society, as well as a collection of actual research, and development activities (Meyer and Rowan, 1977). It is fundamental to the argument of this paper that institutional rules may have effects on organizational structures, managing information and knowledge assets, and especially on different processes in the organization. The main idea is that developed or/and improved processes must be institutionalized in order that they can be incorporated in organization's ways of action, strategy and operating plans. The objective is to get those processes – consisting of all required knowledge and competences (both organizational and individual) - to be confirmed knowledge systems or construction. And like Soin (1999) is saying; *"people may come and go, but processes stay"*.

3 Changing role and position of the management

According to Dr. Deming (Walton, 1994), management has two sets of problems; those of today and those of tomorrow – on the supposition that there is a tomorrow for the company that hopes to stay in business. The problems of today concern the immediate needs of the company: how to maintain quality, how to match output to sales, budget, employment, profits, service, public relations, forecasting. He advises companies to think hard about future, developing both a plan and methods to stay in business.

Peters and Waterman (1982) have developed the profiles of "excellence" organizations. According them, in today's business environments organizations need to organize and manage themselves decisively in new ways, which question old models of thinking and behavior. Organizations have to increase the speed, the rate of change of adaptation and reactivity, so that they learn to develop and change their ways of doing things, for example, change their key processes when needed. There can be seen a new paradigm of management and it's role in business environment, which can be described in three points:

- The change in factors of production. The meaning of traditional factors of production is decreasing, since the utilization of other elements, like intangible assets raise the approval among managers.

- The increasing focus on knowledge assets and individual competences. The importance of knowledge assets, intelligence, individual and organizational competencies as one of the major resources are emphasized. The dilemma is, how, for example, individuals' know-how, performance and creativity are able to transfer forward, to the utilization of the whole organization.

- The dilemma of knowledge management; "it is totally new and you must drop everything else and adopt our nostrum". It's crucial to understand that information and knowledge have always been in organizations, it is nothing new, but managing it creates multidimensional challenges to managers. Grönroos (2000) has said that *"knowledge in itself cannot be managed, instead the systems and processes being contained in knowledge, will be able to manage"*.

The position of the knowledge management cannot be seen as a separate function in organization's operations. Other sources, which will help management to use knowledge management in the developing and improving theirs processes, are, for example, Information Systems, Finance and Accounting, Engineering and R&D, and Human Resources. With the help of Information Systems (includes information sharing), technology infrastructure can be mapped for the use of knowledge management. Finance and Accounting can help figure how to value knowledge and the efforts to manage it, and Engineering with R&D can help mastering particular knowledge, with product knowledge. Human Resources may help to motivate workers to share and use knowledge, and to identify knowledge nodes – individuals, teams, and networks (Davenport and Prusak, 1998).

There are clear signs that companies will move in the next few years towards systematically understanding and improving the management of the new product design and development process while also ensuring it fits with the business activities of the organization (McQuater, Peters, Dale, Spring, Rogerson and Rooney, 1998). By managing all the knowledge, both on individual and organizational level, it's time to find such tools for management, which help to combine different kinds of resources in the organization. These new tools have to be developed to exemplify and identify all this information more quickly and more accurately.

Implications of Motivation and Commitment

Especially human resources play essential part in the new requirements of management. Management must take charge of various kinds of cross-functional teamwork, respond to the demand for continuous improvement, undertake the strategy development processes, and first of all, to achieve not just collaboration but also mutual trust both across functions and also up and down the hierarchy. Particularly the notions of motivation and commitment are emphasized when dealing with change resistance shifting to new ways of management. Juran (1974) defines motivation in the following manner:

> Motivation is the process of stimulating people to act in ways, which serve the needs of the organization providing the stimulus.

It is important that the senior managers "buy" the improvement effort; a full commitment of management is desirable but not decisive (Beer, Eisenstat and Spector, 1990). The committed managers are able to transfer knowledge of the improvement efforts and their results to other, maybe more sceptical managers. Management support for improvement efforts at all levels is critical to the success of the process or product quality improvement effort. Optimally this requires an active involvement of all the managers involved in the selected processes, as well as of the managers who provide input into these processes and receive output from them. (Pastinen, 1998). Management's responsibilities include (adapted from: Harrington, 1991):

- Providing the resources required, including staffing and capital
- Developing common objectives that support the proposed changes
- Breaking barriers between organizations
- Searching out improvement opportunities
- Setting up department improvement teams to support the processes being evaluated
- Changing its own thinking to get a total process perspective
- Providing the necessary training and education to support the new processes
- Anticipating the impact of process changes on their organization and preparing for them
- Establishing systems and reviews to ensure that the progress does not degrade
- Rewarding teams and individuals who make significant contributions to the improvement effort
- Showing interest in the improvement effort by frequent reviews of status results
- Finding equivalent or better jobs for people whose jobs have been eliminated as a result of the improvement efforts

The importance of the role of management is enormous extensive when dealing quality improvement of the development processes. Deming has said that management must lead the way. Only management can initiate improvement in quality and productivity. Improvement is not a one-time effort, so management is obligated to improve continually.

4 The nature and value of knowledge

There is used in knowledge management mainly concepts data, information and knowledge. According to Davenport and Prusak (1998) data is by its nature so-called primary information, structured data record and transactions. It can be said that data is unfinished information, which is typically stored in to the information systems. Data is always a one-dimensional concept, and alone it is not worth nothing, but it is going to be valuable, when it is combined other data and is analyzed together, for example, with project leaders. To same extent, information is more about "active" data, meaning the same than a message – message, which have a impact on receiver's knowledge, know-how and behavior. However, "knowledge" which is held only by people, contains instruction in "how" things are accomplished. This is more complex, more valuable, and also more elusive. Knowledge is arising throughout people's experience in course of time - experience is the result of daily work and provides a historical perspective to view new situations and events. From the base of experience, connections between what have happened before and what is happening now, can be created.

These notions, data, information and knowledge can be seen as a process, from data to information, and from information to knowledge, and this way to know-how. The know-how of organization generates both tangible and intangible capital or substance of organization. Moving from data towards know-how, knowledge is becoming more active and human. In know-how the knowledge is able to apply for the purpose of accomplishing some mission or solving the problem.

Knowledge has no boundaries. Most companies are not using their intellectual resources up to their full potential. Most can be traced to a misfit between required capabilities and those that are available, as the transition from the industrial economy creates new kinds of needs that are only now becoming fully understood. The emphasis has shifted from the focus on managing hard assets, labor, and technology, to managing tacit and explicit knowledge and business processes in a competitive architecture (see figure 4.1). The relative importance of capital and tools has diminished significantly in the Knowledge Age, while people with knowledge, technology, and businesses processes have become much more important.

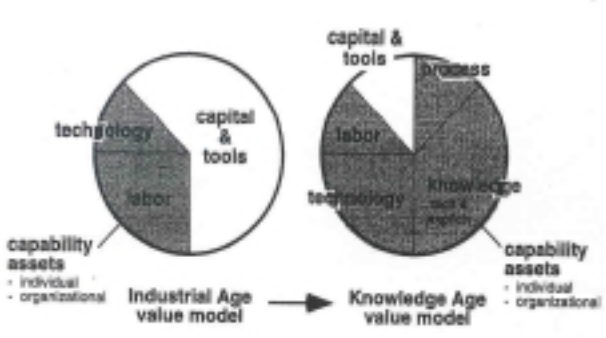


Figure 4.1 Capability Assets in Two Ages (Miller and Morris 1999, 162).

Knowledge can be considered the insights, understandings, and practical know-how that we all possess, it is the fundamental resource that allows us to operate intelligently. The interdependency between knowledge and intelligence is integral, so intelligence creates that world in which it operates, and where knowing occurs (Tuomi, 1999). He is presenting the relations between intelligence, knowledge, and capability as shown in Figure 4.2. Here intelligence generates knowledge structures that underlie capabilities that manifest themselves in selective action. The figure doesn't mean that intelligence, knowledge, capability and action follow each other chronologically. In real life also action can produce knowledge directly (i.e. learning –by-doing or action learning, Argyris and Schön 1996, 50), and develop in this way individual's know-how and capabilities.

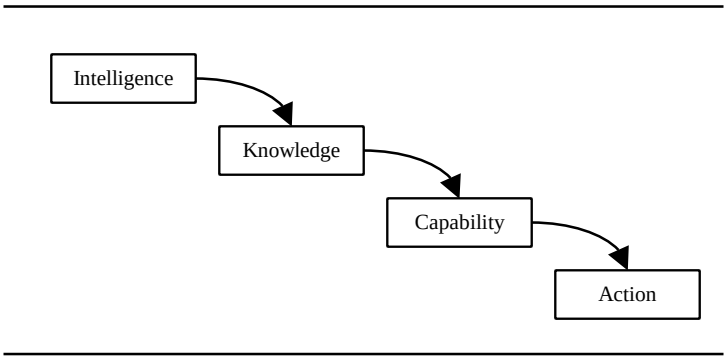


Figure 4.2 Intelligence as generator for selective action (Tuomi 1999, 122).

Nonaka (1991) has stated that *"In an economy where the only certainty is uncertainty, the one sure source of lasting competitive advantage is knowledge"*. The nature of knowledge can be described, for example, time critical, virtual, reflexive, complex, interactive, intuitive, evolving, social, often self-organizing, creative, nonlinear, and selective. Nonaka has with Takeuchi (1995) brought more extensive meaning for tacit and explicit knowledge. It is essential to make a difference between these notions of knowledge, so that we could understand these nature and importance in developing and improving our processes.

- Tacit knowledge is context-specific, personal, and it is hard to formalize and communicate. It is part of everything that we do and say, and as it is inherent in our very thinking, it is deeply embedded in the way that we work. First of all, tacit knowledge is unconscious value.
- Explicit knowledge is easier to identify, because it is written knowledge. It can be stored as artefact – physical or virtual - in different systems, so that it can be identified, measured, distributed and audited. Explicit knowledge is re-usable in consistent and repeatable manner.

Especially in today's competitive marketplace the success of businesses depends critically on the quality of knowledge, which those organizations apply to their key business processes. For example the process development and improving requires knowledge of new technology, information processes, individual and organizational competencies,

organization's strategy, etc. In knowledge assets can be included organization's processes, products, technologies, markets – all the elements that business owns and needs to own and which make possible its business processes to generate profits, add value, etc.

Knowledge can be considered as information combined with experience, context, interpretation, and reflection, says Davenport, De Long and Beers (1998). It is a high-value form of information that is ready to apply to decisions and actions. While knowledge and information may be difficult to distinguish at times, both are more valuable and involve more human participation than the raw data.

Knowledge, as such, has no intrinsic value, and only in relatively exceptional cases we can fix a price tag on a specific piece of articulated knowledge. The value of knowledge depends on a complex social system of activity that creates value using knowledge, and often knowledge transforms into value only at a later time and only for agents that have complementary resources available. The value of knowledge is difficult to estimate because of a fundamental problem: knowledge simultaneously underlies the social division of labor, enables effective action, and is the basis from which ways of working possible. However, even if the value of knowledge is something we cannot know in general or absolute terms, we still need to be able to measure organizations in the knowledge dimensions. (Tuomi, 1999)

5 Managing knowledge

There can be found in many organizations that existing knowledge is not fully exploited by the management. It seems that organizations do not complete understand how to take advantage of all that competence and knowledge within organization. Management should also pay more attention to the utilization of process improvement, together with of quality and competence approaches and methods. A presupposition for successful implementation of improvement approaches is the use of appropriate knowledge.

Especially the need for the development of high-class competencies, involving both on individual and organizational level, is extensive. There is needed effective strategic management of an organization's knowledge assets, thus, considered both from individual and organizational point of view. It requires recognition of the potential strategic value of each of the organization's different stocks of knowledge. Ron Sanchez (1997) proposes a framework for analyzing an organization's knowledge assets that suggests several approaches to leveraging and controlling an organization's strategic knowledge assets. After a critical appraisal of the strategic value of "tacit" knowledge within organizations, he suggests that knowledge within companies has different *contents* – which are characterized as *know-how, know-why,* and *know-what* –that are used in different strategic *contexts*.

It is obvious that KM has significant implications for TQM and continuous improvement processes and strategies. Even though knowledge management is somewhat problematic its nature, current perspectives on knowledge management can still be characterized as extensions of either information management systems, which can consider the IT paradigm, or organizational learning, which ally more into humanistic perspective.

When discussing about organization's development processes, process improvement techniques[3] come up. A successful implementation of process improvement requires also marketing and business knowledge, psychological and communication skills as well as a true spirit to improvement including the critical examination of the own performance (Pastinen, 1998). It is important to understand that using only one approach, technique or method will not be enough in today's quality improvement of the processes.

Different kinds of process improvement techniques are needed, but these must not regard as a universal list for development of processes. Because organizations are operating in fast changing markets, it is prerequisite to develop also these process improvement tools and methods. It is crucial to remember that in the use of all these tools is needed human being, intellectual individual whose knowledge and know-how must be able to utilized. It is needed flexibility and new management of knowledge, so that we can keep up with competition.

Knowledge Management Process

Knowledge engineering methods and tools have come a long way towards addressing the use of a company's knowledge assets. They provide disciplined approaches to designing and building knowledge-based applications. There are tools to support capture, modeling, validation, verification and maintenance of the knowledge in these applications. However these tools do not extend to supporting the processes for managing corporate knowledge (Macintosh, 2000).

I'm presenting a knowledge management cycle, one way to manage knowledge by pointing, what main issues have to take into account when using knowledge management in developing and improving quality in processes. There can be found some similar factors and ideas with other improvement cycles, for example Deming Cycle ('PDCA Cycle' because it was Dr Deming who introduced it), or with knowledge processes, like the learning cycle of Nonaka and Takeuchi (1995). I'm describing four stages to manage knowledge, and also what processes are in these stages included. The stages for knowledge management are presented in Figure 5.1, which would call as "Knowledge Management disk".

The first step for knowledge management is *identification*. On this stage is identifying the need, urgency and form of necessary knowledge assets. Here are many problems associated with finding out organization's knowledge assets and being able to use them in an efficient and cost-effective manner. At first, it is essential to map out, how acute need there is for process development, and also for the implementation of these processes. When also the need and urgency of knowledge has been specified, its time to find out what kind of knowledge is needed and where are they located within organization.

[3] Considering quality management as an example, there is a vast number of approaches advocating a step-by-step approach to process improvement. Examples of such include Deming's universal fourteen points to management, Juran's ten steps to quality improvement, Crosby's fourteen points, Vendelø's CASE tools and Beer's, Eisenstat's and Spector's "Critical Path" (a six-step approach).

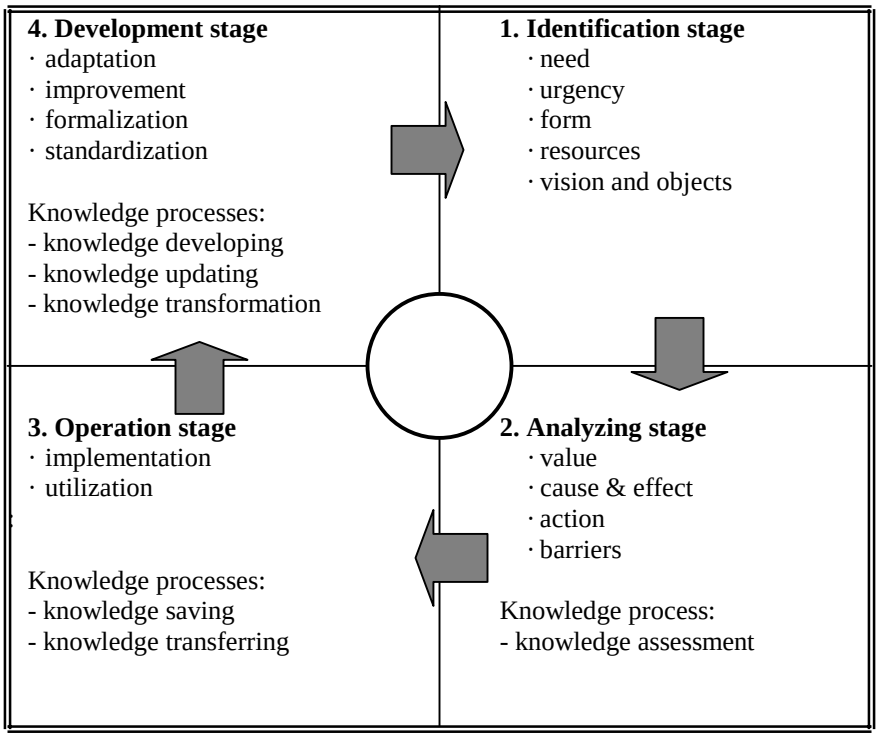


Figure 5.1 Four stages for knowledge management – "Knowledge management disk".

Knowledge can be found from very different places and concerns within organization. In this can be related the concept of organizational memory, therefore the metaphor and construct of organizational memory has multiple interpretations, and its use varies across research traditions. For example, organizational memory can be associated with two kinds of data, "hard data" and "soft data". Hard data consists, among other things, databases, software, risk tools, audits, reports, manuals and policy documents. Soft data is more associated with individuals, theirs competencies, ability to learn new things, peoples know-how and experience. Besides soft data is stored in peoples' capabilities, it can be traced also from both internal and external social coalitions or clusters in the organization, for example, teams, groups, communities and networks.

When we are identifying these knowledge resources in diversified sources, it is crucial to find out and understand what kind of information these knowledge assets contain, and what purpose they are able to use. By specifying the form of knowledge, and its most optimal practical application, it is possible to achieve better final outcome, and this way – of course – the best satisfied customers. In the very end, customer values the most how this output serves its needs, and what is its utilization and added value for its operations. Organization must take care of that it can offer the best possible applications via the best possible knowledge and development processes.

Another important issue, which is included in identification stage, is to create the vision and set the target and objectives. Vision answers the question where are we going, giving the direction of the upcoming improvement efforts. Objectives include more detailed statement

of the item, it can be either numerical (i.e. improve output to x percent by year xxxx) or other measured value, like improved quality. Setting target must be reasonable and realistic but not too easily attainable. Objectives also describe more detailed the ways and means how to get to vision. Both vision and objectives should have mutual support and commitment within organization, both with management and employees. It is important to "visualize" these points in every level of the organization.

On *analyzing stage* is studied how these – on the previous stage identified - knowledge assets can add value, and what are the opportunities for using the knowledge assets. Like I earlier mentioned, knowledge is a high-value form of information that is ready to apply to decisions and actions. The value of knowledge involves much more human than the raw data - it depends on a complex social system of activity. The importance of individuals cannot underestimate, because they are bringing theirs capability, intelligence, know-how in the use of organization. Blackburn, Scudder and Van Wassenhove (1995) have said that talented people are essential to a fast development process. Virtually no amount of management technique and team organization can overcome a lack of talented designers and coders. This is not a new observation, but is a recurring theme in the literature on innovation, research and development. The lesson for managers is that, since these talented people are so important to the time-to-market process, the firm should make a special effort to identify, reward and make heroes of their best people. Making most effective use of development talent must be a key concern for software managers

In analyzing organization's knowledge assets, must pay attention to what would be the cause, and also the effect of theirs use. By mapping the whole chain of knowledge from the need to the effect of this knowledge, leads us to better understanding what is the eventual advantage and utility to use particular form of knowledge. Besides we know what is the true value of knowledge, we must also ask what would be the increased value of knowledge to the organization. In order that organization will have the most valuable knowledge for its utilization, it is substantial appraise what sort of action should carry out, and what current and potential barriers there exists.

Especially the knowledge process, assessment, is associated in this stage for knowledge management. These above-mentioned issues should appraise, evaluate, validate, verify, etc. the most accurately way.

Through the third stage, *operation stage*, organization has possibility to achieve the identified objectives. Correspondingly, the main objectives of this stage are to implement and utilize the identified and analyzed knowledge, to be able to gain the best possible result. First, the analyzed and required knowledge must collect, pack and preserve in the right format, in order that it is easier to formalize in specified function. After that, the knowledge must be secured, so that organization can rely on its constancy and stability. Being "unprotected", there is a risk that this knowledge is changed, or even is vanished or destroyed, for example because of unexpected changes in organization's information channels, technologies, or even in business environment.

The next thing is experienced probably the most difficult phase to perform. Transferring the knowledge contains various functions, for example communicating, deploying and sharing

the knowledge. The knowledge must be able to transfer within organization, between organization levels, projects, individuals, etc. Japanese professors Nonaka and Takeuchi (1995) have provided a major contribution to the theory and practice of knowledge management. Much of the recent interest in knowledge management can be traced back to theirs work. To visualize how to share and transform tacit and explicit knowledge, Nonaka and Takeuchi have been developed a matrix that describes the transitions of knowledge (Figure 5.2). The matrix shows that tacit knowledge can be shared from one person to another without being made explicit, the process of **socialization** is used in advertising to convey social meanings that are powerful, even as they are intended to remain at the unconscious level. This socialization process happens through observation, imitation, practice, and shared experience.

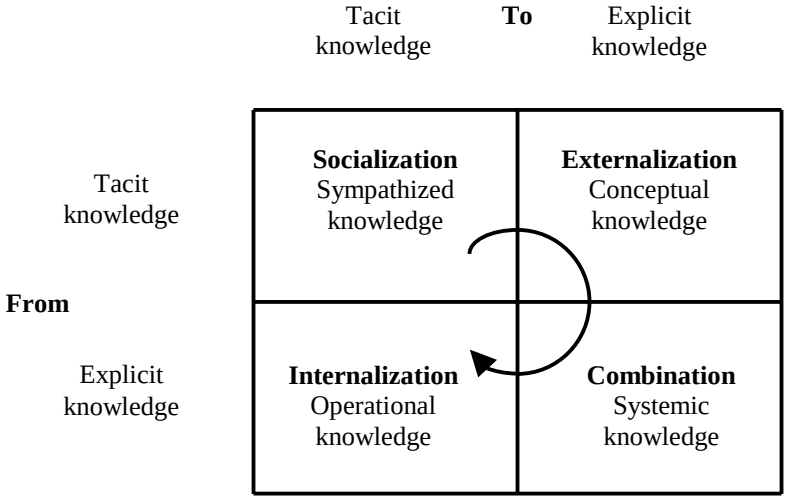


Figure 5.2 The transitions of knowledge (Nonaka and Takeuchi, 1995).

The upper right quadrant shows that when tacit knowledge is made explicit, it is **externalized**, made manifest in spoken words, writings, or tangible objects. Researchers seek to do just this, rendering the hidden tacit forms explicit, and therefore applicable in the innovation process. In this process, tacit knowledge takes the shape of metaphors, analogies, concepts, hypotheses, and models. It can be said, that externalization holds the key to knowledge creation, because it creates new, explicit concepts from tacit knowledge.

Combination occurs when explicit knowledge is shared and integrated through learning. Combination process includes such activities as sorting, adding, and categorizing explicit knowledge, for example, in formal education and training at schools. Explicit knowledge is made tacit when it is **internalized** through experience. Experiences through socialization, externalisation, and combination are “internalized into individual’s tacit knowledge bases in the form of shared metal models or technical know-how”, and therefore become valuable assets.

This model points out that as knowledge moves through an organization or a society, it often moves from one quadrant to the next as indicated by the spiral that connects the four

quadrants. It can be said, that innovative learning and knowledge creation is understood as conversion of tacit knowledge into explicit forms where it can be combined, followed by an internalization process where this new combined knowledge becomes a part of the learner's knowledge structure.

In this knowledge discuss has come up, that in a knowledge economy, companies that could leverage their knowledge and transform them into competitive advantage, will advance. This means that companies should invest heavily in knowledge that could increase productivity and innovation.

On the fourth stage, on *development stage*, knowledge is further adapted. Knowledge developing process contains, for example, capturing the required knowledge, to create, and find the new forms of knowledge for the next processes, when the knowledge is needed. Knowledge must also update, meaning improving, evolving and also maintaining the knowledge. So that knowledge is able to maintain continuously, it is crucial to review how the certain kind of knowledge will ensure added value and ask have we achieved the desired added value of quality?

The main idea of this paper is culminating in next knowledge process – knowledge transformation – where the knowledge is compiled, formalized, and standardized. Knowledge assets need to be "classified by type", to allocate them in certain categories or forms, and this way to create some kind of "hierarchy of needs". By formalizing knowledge assets by the need (i.e., in what purpose the knowledge is needed, how soon, who needs it, etc.), it is easier and more effective to address the exact information in the certain need. The formalization and standardization of knowledge assets makes the utilization of these assets much more efficient.

It is fundamental to the argument of this paper that institutional rules may have effects on organizational structures, managing information and knowledge assets, and especially on different processes in the organization. Different ways to operate and behave are defined by various regulations and rules of orders. Because organization's all functions include some sort of information and knowledge, it is obvious that through standardization of these knowledge assets the management is able to get the situation where organization's knowledge or intellectual capital are available, ready for the new purpose and use. So, knowledge management can be seen as a cycle where different knowledge stages can be repeated even on several occasions during the development processes.

Conclusions

Knowledge management can be seen as an essential capability in the today's emerging knowledge economy. Research suggests organizations to use knowledge management in circumstances where their needs and requirements change rapidly. Knowledge management process/system - which organization is using – have to have a good reactivity, and also to function, take action explicitly in the lead of changes. It must be able to predict those sudden and sometimes very revolutionary changes happening in business environment, and not just wait them happen.

The one primary dilemma in Knowledge Management is that it is hard to make concrete and that way to manage it in comprehensive way. There can also be found some difficulties in commitment, how to commit KM to organization's activities, procedures, and/or operations models, but also, how to manage people's knowledge, know-how and intelligence – intellectual capital. It is important to notice that knowledge management draws from existing resources that your organization may already have in place – good information systems management, organizational change management, and human resources management practices (Davenport and Prusak, 1998). The basis of the information and knowledge can be found within organizations, but in the very end, organization itself has to make its own decisions about what knowledge is most crucial to manage, how to motivate people to share and use knowledge, and what will make a process or project succeed in its own specific environment.

One of the basic tenets of total quality is continuous improvement. It can be considered that in any organization, we need both improvements and breakthroughs. Managing organization's key processes and improving them will help to increase efficiency, productivity, and quality. The results will be a more competitive and productive personnel that provide increased customer satisfaction, higher profits and commitment. Especially, Deming (Walton, 1994) has emphasized that statistical thinking is critical to improvement of a system. Only by use of properly interpreted data can intelligent decisions be made. But to depend only on the use of statistics is a sure way to go out of business. But after all, the exact knowledge have to be available, the right, useful, just-in-time, and 'just-in-need' (to be focused in certain need) in order that, organization will get the fully benefit. Development process should be seen as an approach to improve and increase organization's performance.

Organizations must rely on information and knowledge tools to coordinate and concentrate knowledge flows within organization, between units, departments, teams, accordingly both on individual and organizational levels. Development processes – which can been considered as one of the most important activities in improving the quality – should not carry out "slavishly", "by the books", but using these levels and phases as tools to develop the organization's own processes. There must take into account organization's own needs and competences, especially both existing information, knowledge, know-how and its management, but also pay regard to the creation of new knowledge.

The objective of this paper is not to give an integral analysis of conceptual system how to manage all that information and knowledge within organization, but to create a one perspective to review the knowledge and its value in development processes – both on organizational and individual levels. When KM is used to improve quality in all stages of organization, it is important to know and understand what kind of knowledge processes exist within organization. As I earlier mentioned, the main idea is that developed or/and improved processes must be institutionalized in order that they can be incorporated in organization's ways of action, strategy and operating plans. The objective is to get those processes – consisting of all required knowledge and competencies (both organizational and individual) - to be confirmed knowledge systems or construction.

REFERENCES

Amidon, C.M. (1996). The Momentum of Knowledge Management. Research/Technology Management, 39 (4, May-June).

Barley, S.R. (1986). Technology as an occasion for structuring: Evidence from observations of CT scanners and the social order of radiology departments. Administrative Science Quarterly, 31, pp. 78-108.

Beer, M., Eisenstat, R. and Spector, B. (1990). The Critical Path to Corporate Renewal. Harvard Business School Press.

Blackburn, J., Scudder, G. and Van Wassenhove, L.N. (1995). Improving Speed and Productivity of Software Development: A Survey of European Software Developers. INSEAD Working Paper, Fontainebleau, France.

Boutellier, R., Gassman, O. and von Zedtwitz, M. (1999). Managing Global Innovation. Heidelberg, Berlin.

Davenport, T.H., and Prusak, L. (1998). Working Knowledge: How Organizations Manage What They Know. Harvard Business School Press, Boston, MA.

Davenport, T.H., De Long, D.W. and Beers, M.C. (1998). Successful knowledge management projects. Sloan Management Review, (Winter), Cambridge.

DiMaggio, P.J. and Powell, W.W. (1983). The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields. American Sociological Review, vol. 48, April, pp. 147-160.

Dutta, S., Kulandaiswamy, S. and Van Wassenhove, L.N. (1996). Benchmarking European Software Management Best Practices. Working Paper, INSEAD, Fontainebleau, France.

Gannon, P. (1997). Trojan Horses & National Champions: The Crisis in the European Computing and Telecommunications Industry. Apt-Amatic Books, London.

Gore, C. and Gore, E. (1999). Knowledge Management: the way forward. Total Quality Management, vol. 10, no. 4-5, pp. 554-560.

Greenwood, R. and Hinings, C.R. (1996). Understanding Radical Organizational Change: Bringing Together the Old and the New Institutionalism. Academy of Management Review, vol. 21, no. 4, pp. 1022-1054.

Grönroos, M. (2000). “Knowledge Management” –seminar on May 4th 2000. FINESSE SPI, VTT, Oulu.

Haldin-Herrgård, T. (1999). Difficulties of Tacit Knowledge Management – A Study on Diffusion of Tacit Knowledge in Organisations. ISO 9000& TQM for 2000+.

Harrington, H.J. (1991). Business Process Improvement. McGraw-Hill Inc., New York.

Hinings, C.R., Greenwood, R., Brown, J. and Cooper, D. (1996). Organizational Change: the Role of the Archetypes, Environmental Dynamics and Institutional Ideas. In production Global Perspectives on Prosessual Research on Management and Organization, edited by A. Ropo, P. Eriksson and J. Hunt. University of Tampere, series C: Conference Papers and Occasional Papers 6, Tampere, pp. 41-75.

Jones, M.R. (1990). Ecuivocality and Information Systems. Working Paper, Cambridge University, Engineering Department.

Juran, J.M. (1974). Quality Control Handbook. Third edition, Mc-Graw-Hill, New York.

Leonard-Barton, D. (1995). Wellsprings of Knowledge. Building and Sustaining the Sources of Innovation. Harvard Business School Press, Massachusetts.

Macintosh, A. (2000). Knowledge Management. Position Paper on Knowledge Asset Management. [WWW-document]. <http://aiai.ed.ac.uk/~alm/kamlnks.html>.

Maxwell, K., Van Wassenhove, L. and Dutta, S. (1996). Software Development Productivity of European Space, Military and Industrial Applications. Research Initiative in Software Excellence (RISE). INSEAD, Fontainebleau, France.

McQuater, R.E., Peters, A.J., Dale, B.G., Spring, M., Rogerson, J. and Rooney, E.M. (1998). The Management and Organisational Context of New Product Development: diagnosis and self-assessment. International Journal of Production Economics, vol. 55 (2).

Meyer, J.W. and Rowan, B. (1977). Institutionalized Organizations: Formal Structure as Myth and Ceremony. American Journal of Sociology, vol. 83, no. 2, pp. 340-363.

Miller, W.L. and Morris, L. (1999). 4th generation R&D : managing knowledge, technology, and innovation. John Wiley & Sons, Inc., New York.

Nonaka, I. (1991). The knowledge-creating company. Harvard Business Review, (November-December).

Nonaka, I. and Takeuchi, H. (1995). The Knowledge-Creating Company : How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, Oxford.

Pastinen, M. (1998). Process Improvement Essentials: A Framework for Creating and Implementing Operational Improvement Plans. 2nd edition, Vistalize Ltd., Helsinki.

Peters, T. and Waterman, R.H. (1982). Menestyksen profiileja. Hyvin hoidettuja yrityksiä ja mitä niistä opimme. Original book: In Search of Excellence. Rastor, Helsinki.

Sanchez, R. (1997). Strategic Learning and Knowledge Management. In R. Sanchez and A. Heene (Eds.), John Wiley & Sons, New York.

Scott, W.R. (1995). Institutions and Organizations. Sage, Thousand Oaks CA.

Soin, S.S. (1999). Total quality essentials: using quality tools and systems to improve and manage your business. McGraw-Hill Companies, New York.

Tuomi, I. (1999). Corporate knowledge. Theory and Practice of Intelligent Organizations.

Vendelø, M.T. (1998). Recycling software – on the road to high performance in software companies. International Journal of Technology Management, vol. 16, Nos. 1/2/3, pp. 93-104.

Walton, M. (1994). The Deming Management Method : The Complete Guide to Quality Management. Management Books 2000 Ltd., Chalford.

Weick, K.E. (1990). Technology as equivoque: Sensemaking in new technologies. P.S. Goodman, L.S. Sproull, and Associates, Technology and Organizations, Jossey-Bass Publishers, San Fransisco.

QWE2000 Vendor Technical Presentation VT12

Mr. Bob Bartlett
(SIM Group)

The dream comes true - Scriptless Automated Testing

Key Points

- Record play back techniques do not produce well-designed automated tests that can be re-used and grown for continuous automated testing. At best, record & playback will give the ability to play back the same tests for very few test executions. There is no substitution for well-designed tests that have flexibility, robustness, reusability and expandability. However, producing automated tests with these characteristics is time consuming and requires high levels of programming skills.
- A well designed automated test system is developed using software engineering disciplines to support a wide variety of tests that survive a number of changes and evolutions of the system under test. This design practice should be applied and maintained as testing requirements are defined, matured and evolve.
- The test system must also support good practices and efficiencies in test planning, test case preparation, test execution and problem management.
- The way to achieve the foregoing requirements and allow non-programmers to develop and run automated tests is to use an extreme implementation of table driven testing. SIMÆs table driven testing technique and software (tMosaic) satisfies all of the requirements of good automated testing design and practice, but is controlled and used by professional testers that do not possess programming skills.

Presentation Abstract

o The dream for automated testing – what we all want.
o Why record and play back techniques have not satisfied the requirements.
o Test System Design
o Table Driven Testing
o The SIM methodology for Table driven testing = tMosaic
o How this approach makes the dream come true

About the Speaker

Bob is the Chairman of SIM Group Ltd. SIM specialises in Software Testing and has put in place a number of highly efficient testing systems that automatically test sophisticated and mission critical software systems. SIM is the UK leader in

providing efficient solutions for software testing. SIMÆs work has had a profound impact on the way companies approach testing and improvements to testing have been realised with SIMÆs help.

SIM has managed the development of testing strategies for software projects and has implemented automated testing techniques for many different software environments.

A summary of BobÆs experience follows:

- Over 30 years in software, and using automated testing techniques throughout
- Executive Director and Chairman of Software testing specialist company today
- Member of the CSSA executive council
- Has designed, developed and sold automated testing tools
- Manager of major software development and implementation projects
- Testing adviser to some of the largest testing projects taking place in U.K.
- Training and lecturing in automated testing and software testing techniques
- Track record for substantial reductions in time and cost to test
- Successfully managed the growth of start up companies throughout his career.

GRESHAM

Scriptless Automated Testing

Bob Bartlett
Managing Director
bob@simgroup.co.uk
www.simgroup.co.uk

SIM GROUP

Agenda

GRESHAM

- ***The record / playback myth***
  - Why it can't work as advertised
- ***The dream for automated testing***
  - What everyone really wants
- ***The first step of good practice***
  - Test System Design
- ***Data Driven and Table Driven Testing***
  - Advantages and ideals

SIM GROUP

Agenda Continued

GRESHAM

- Test System Framework
- tMosaic & SIM's methodology for automated testing
- Example of benefits

SIM GROUP

Background...

GRESHAM

Who are we?

- Independent testing organisation
- Specialists in testing for 9 years
- Largest implementor of automated testing in UK
- Full Service testing Solutions

What do we do?

- Testing Projects
- Testing Improvement
- Implementors of automated testing for over 11 years
- Methods, procedures and strategies for efficiency & effectiveness
- 100% success on all automated testing work

SIM GROUP

The Record / Playback Myth

GRESHAM

- ***If Record / Playback worked, we wouldn't have scripting languages, wizards or extensive script support:***
  - Only the very oldest of testing tools did not have scripting support
  - Many enhancements to testing tools try to make record / playback work properly!
  - Some of the best tools started life without record / playback
- ***Automating unstructured and poorly designed tests is only "automating the chaos" of testing***
- ***It is only used effectively during sales demonstrations***

SIM GROUP

Experiences With Record / Playback

GRESHAM

- ***"The tests would not rerun because dependent tests or actions had not been run the same."***
- ***"Interfering or asynchronous activities prevented the tests from re-running."***
- ***"The tests were not worth re-running because what was recorded was mostly irrelevant tests."***
- ***SIM's experience***
- ***When tests are recorded, any play back of tests works AT BEST for 50% or the resultant tests - each time playback is attempted.***

SIM GROUP

Why Record / Playback Is Bad Practice

GRESHAM

- ***Test are <u>not necessarily</u> well designed and generally lack:***
  - Re-runnability
  - Robustness
  - Results verification
  - Expandability
  - Flexibility
- ***Time spent fiddling with the tool, detracts from the advantages of manual testing***

SIM GROUP

Automated Testing - What We All Really Want

GRESHAM

- ***For a small minor increase in time to build the automated tests - substantial reductions in time when the tests are run automatically***
- ***Automated tests should be built by testers - NOT software developers***
- ***If one automated test fails, the rest should run without impact from the failing test***
- ***Easy and fast to select tests and run***
- ***Tools should be intuitive to pick up and use when required***
- ***Enhancing the tests for SUT changes or new tests should be fast and easy to undertake***

SIM GROUP

Characteristics of Automated Tests

GRESHAM

- *It should be easy to see the relation between a manual test and its automated version*
- *Easy to build, maintain, modify and execute*
- *The user interface to the tool should be intuitive and easy to use at any time*
- *Tests should be robust, always re-runnable and easily expanded*
- *Tests should run without ANY manual intervention*
- *Tests results, testing metrics and success or failure of tests should be instantly available*

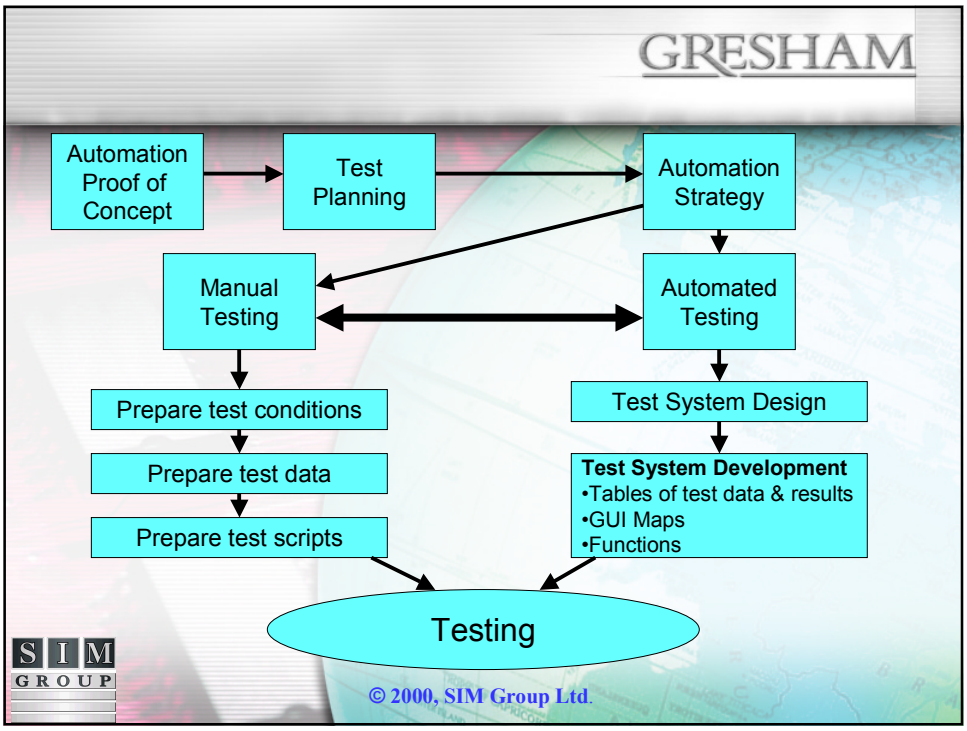
SIM GROUP

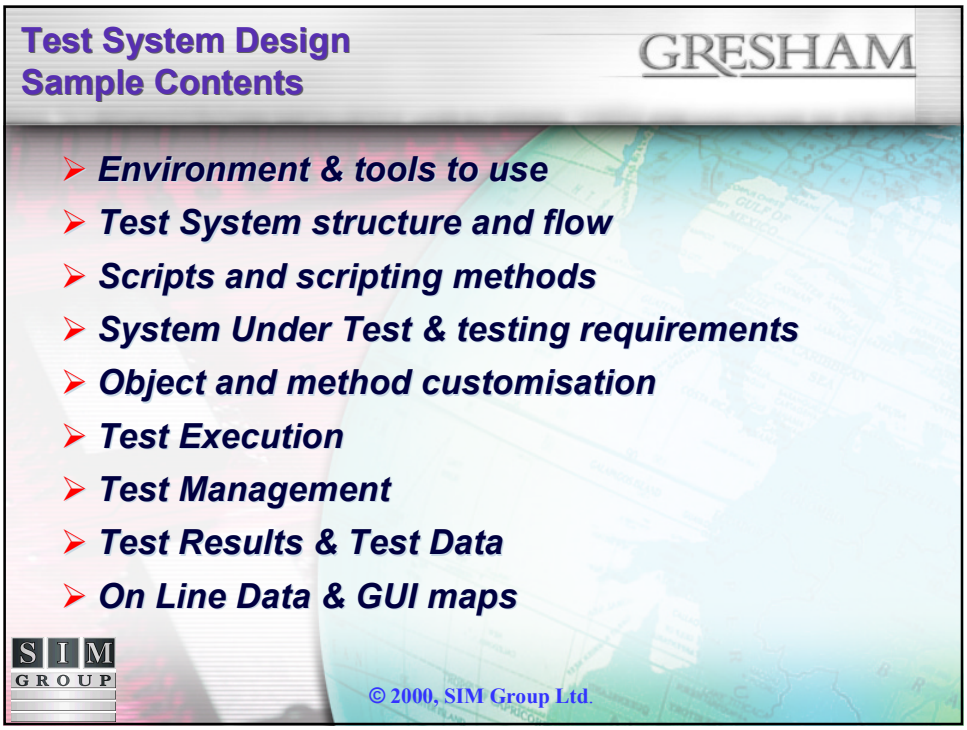
The Implication for Automation

GRESHAM

- *Good Tests are well planned, well designed and thoroughly prepared*
- *The required characteristics of tests come about from the test design process used*
- *The "end users" of automated testing must also be considered and their requirements included.*
- *Automated tests have:*
  - Inputs - conditions, data & expected results
  - Outputs - testing results, metrics, information about the testing
- *Automated Tests ARE a SYSTEM that automates the testing required*

SIM GROUP


GRESHAM
Automation Proof of Concept
Test Planning
Automation Strategy
Manual Testing
Automated Testing
Prepare test conditions
Prepare test data
Prepare test scripts
Test System Design
Test System Development
•Tables of test data & results
•GUI Maps
•Functions
Testing
SIM GROUP
© 2000, SIM Group Ltd.


Test System Design
Sample Contents
GRESHAM
Environment & tools to use
Test System structure and flow
Scripts and scripting methods
System Under Test & testing requirements
Object and method customisation
Test Execution
Test Management
Test Results & Test Data
On Line Data & GUI maps
SIM GROUP
© 2000, SIM Group Ltd.

Implementing the Test System Design - the HARD Way

GRESHAM

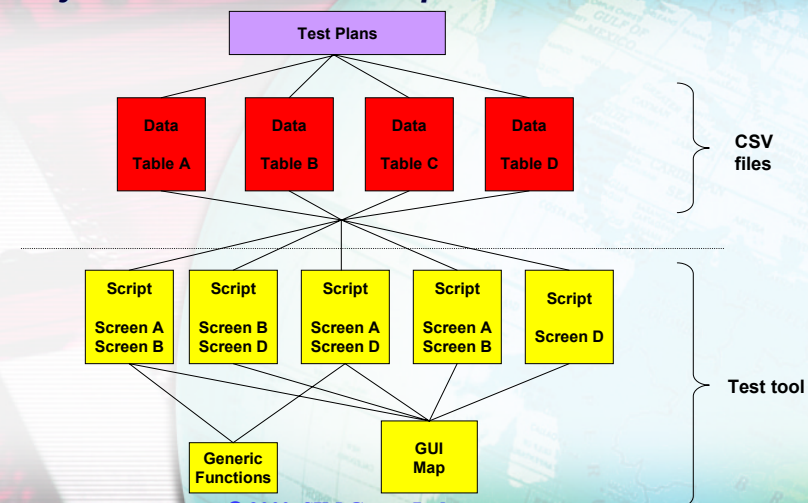
- *The tool MUST work for the SUT and environment*
- *Modular scripting*
- *Develop all requirements*
- *Thoroughly test infrastructure*
- *Design and populate tests*
- *Prove tests and test data*
- *Run many times to make sure it works*
- *Document and train users*
- *Keep test system developers available*

SIM GROUP

Automated Test System

GRESHAM

- *Test System Structure example*



SIM GROUP

Test System Design

GRESHAM

- Sample of data driven testing

	A	B	C	D	E	F	G	H
1	Client_type	Gender	First_Name	Suffix	Surname	Postcode	Day_Phone_Number	Eve_Phone_Number
2	P	M	Mo	Mr	Gil	gu22 7pj	0141 203 1234	0121 376 4876
3	P	F	So	Miss	Tia	SW11 3lp	01823 46576	
4	P	f	Bo	Mrs	Pat	EH11 3LP	04653 78346	01824 786543
5	P	F	Too	Mrs	Ann	le11 3lp	07854 78654	
6	P	F	Koo	Miss	Judith	wa42lp	01245 875632	01824 987543
7	P	F	Sanders	Mrs	Jane	CM6 2QF	01245 764523	0986 922638
8	P	F	Dixon	Mrs	Ann	CM19 5JW	01245 563348	
9	P	F	Bristow	Mrs	Judith	CM20 3JX	01245 465783	0876 987632
10	P	F	O'Brien	Miss	Anne	CM22 6SG	01245 786543	0832 987643
11	P	F	Chalmers	Ms	Julie	CM2 8RF	01245 125677	
12	C		Patts Ltd			NR7 0SE		0376 956874
13	C		Dixons Ltd			NR15 1QJ	0456 876541	
14	C		Pats Trading Ltd			NR10 5AX	0876 987654	
15	C		Argos Ltd			NR21 8BW	0458 874321	0453 675423
16	C		Travis Perkins			NR13 4RH	01456 978543	
17	C		Lunn Poly			CM22 7RL	01245 675487	0976 345678
18	C		Zakz			CM13 2SE	01245 675432	0873 456784
19	C		Davis & Davis			CM15 8AA	01245 564321	0974 965476
20	C		R Dyas Ltd			CM13 1AA	01256 634546	
21	C		Peverstrand Ltd			CM20 2RH	01345 653423	0234 456743

Implementing the Test System Design - the EASY Way

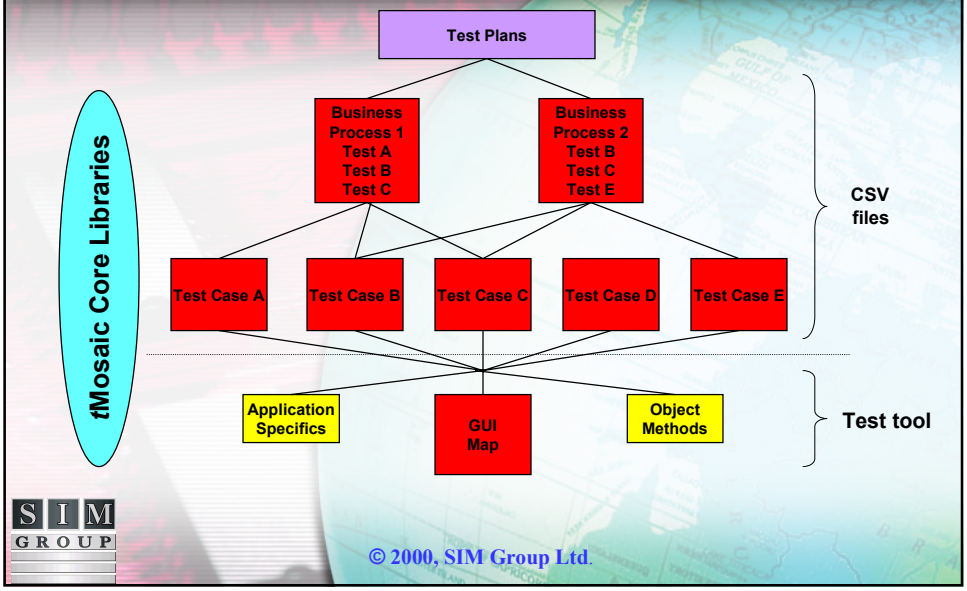
GRESHAM

- *Start with a solid frame work*
- *Customise the planning interface for the SUT and the user*
- *Design the planning structure & prove it*
- *Populate with tests and test data*
- *Prove the tests*
- *Use it*

SIM GROUP

Test System Overview

GRESHAM



SIM GROUP

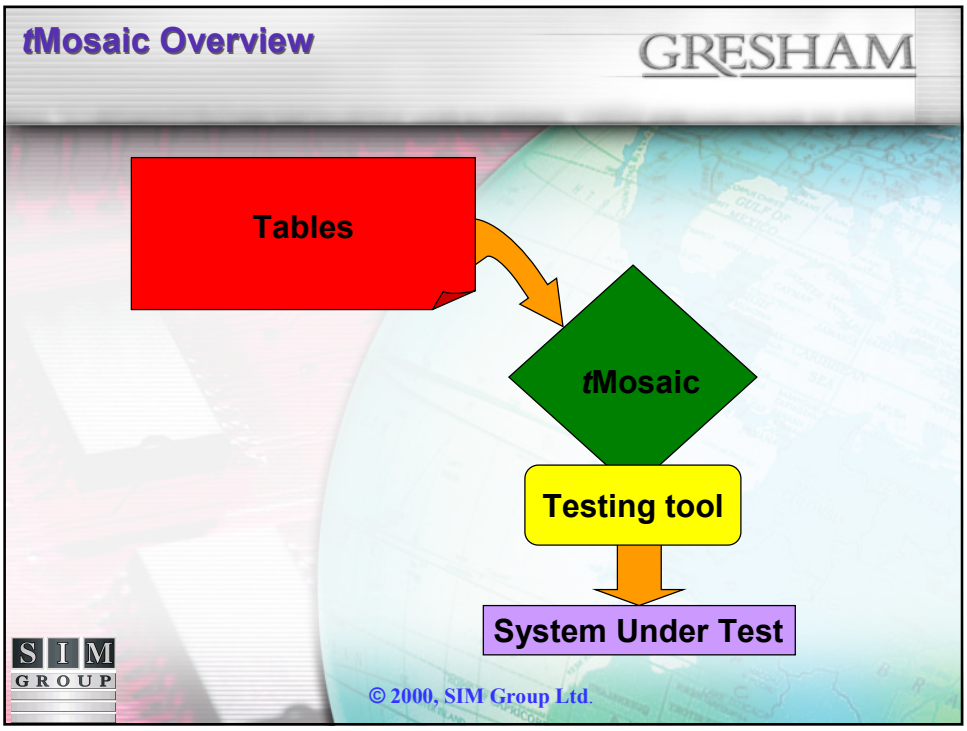
tMosaic Overview

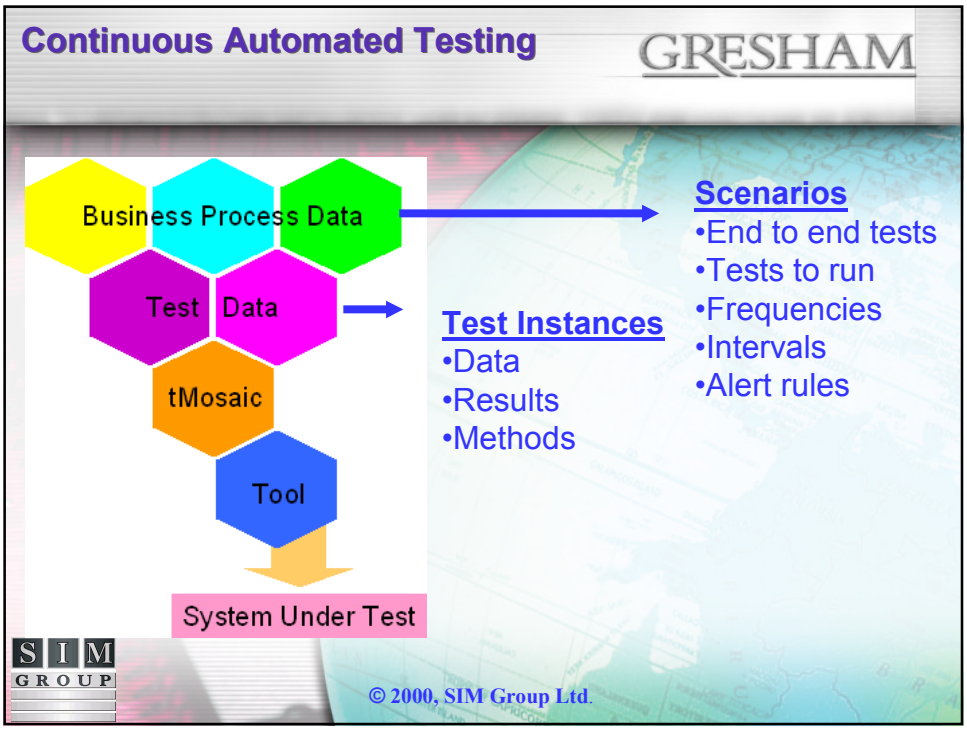
GRESHAM

- *Sample of table driven testing*

	A	B	C
1	Method	Object	Data
2	Open	Flight_Reservation	10
3	Press	New_Order_Icon	
4	Input	Flight_Date	12/12/99
5	Select	Flight_From	Paris
6	Select	Flight_To	London
7	Press	Flights_Button	
8	Open	Flights	10
9	Select	Flight_List	13307

SIM GROUP


tMosaic Overview
GRESHAM
Tables
tMosaic
Testing tool
System Under Test
SIM GROUP
© 2000, SIM Group Ltd.


Continuous Automated Testing
GRESHAM
Business Process Data
Test Data
tMosaic
Tool
System Under Test
Test Instances
•Data
•Results
•Methods
Scenarios
•End to end tests
•Tests to run
•Frequencies
•Intervals
•Alert rules
SIM GROUP
© 2000, SIM Group Ltd.

Experiences & Benefits

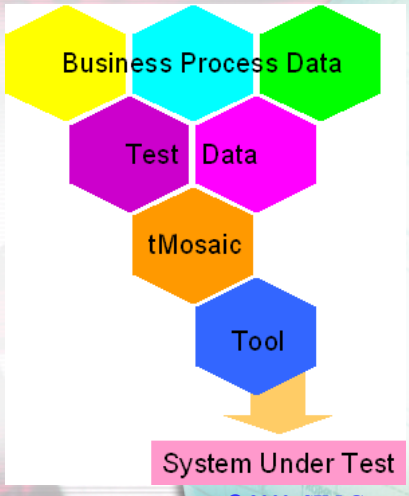
GRESHAM

- *The best testers can now be the best automated testers without having to be programmers as well!*
- *Initial test systems are developed in less than half the time it use to take*
- *Fully robust and unattended automated test systems built in about a third of the time*
- *Testing tables can be developed and proven before the application is available*
- *Test Systems are easy to maintain, support and expand*

SIM GROUP

tMosaic

GRESHAM



SIM GROUP

Bob Bartlett
bob@simgroup.co.uk
www.simgroup.co.uk

QWE2000 Extra Session

Mr. Leif Balter [Sweden]
(Cap Gemini Ernst & Young)

"Create Your Own Testtool"

Key Points

- The power of Excel
- The power of doing it yourself
- Commercial test tools are not always the best choice

Presentation Abstract

With Excel you have the power to do almost anything. You don't have to buy it - everyone has it on the PC, you don't have to go to training classes - so many people already knows Excel. You can make VB macros, you can make graphs and reports and paste it into Word documents.

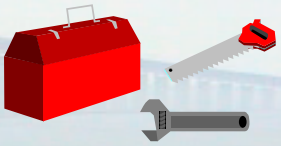
Comericial test tools have always had an attraction to me but after working in y last project surrounded by some Excel experts, I strongly recommend all Project Managers and Test Managers:

Check out the capabilities and possibilities with Excel before introducing any other Test Administration tool.

About the Speaker

Leif Balter has started as Cobol programmer 30 years ago. For the last 10 years he worked at Cap Gemini Sweden. Mostly as Test Manager or Project Manager. He has been deeply involved in building competence networks for software testing, writing articles and as teacher.

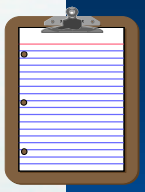
Create your own testtool



- **Leif Bälter,**
  - Cap Gemini Ernst & Young, Sweden

CAP GEMINI ERNST & YOUNG

Agenda



- Typical SW dev startup factors
- The case (Project background)
- Test specifications and statistics
- Defect Reports and statistics
- Test Data generation
- Expected Result calculation
- Used Excel features
- Limitations, Benefits
- Lessons learned

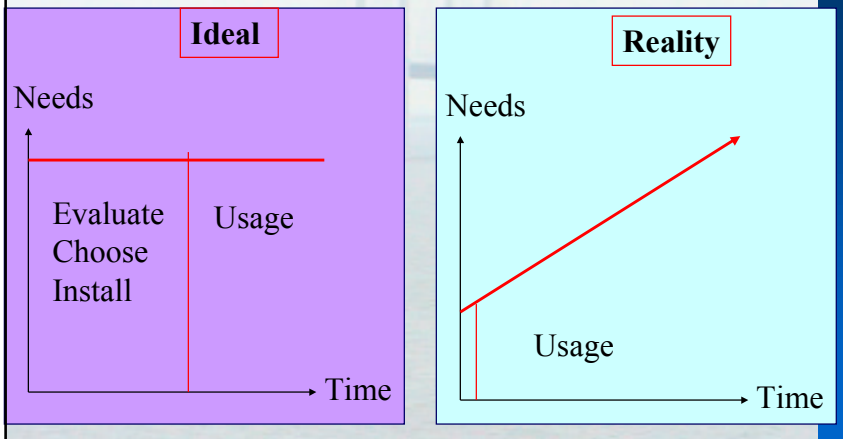
CAP GEMINI ERNST & YOUNG

Some typical SW Dev project startup factors

- Meeting strangers who become colleagues with different experience levels and ages
- Delivery on time important
- No extra budget for tools
- Tools used by PM are treated with special favour
- No time for test tool evaluation
- You always have MS Office including Excel installed

CAP GEMINI ERNST & YOUNG

Test tool requirement



CAP GEMINI ERNST & YOUNG

The project background

- A part of the new Swedish pension system
- Selection of stockbroker funds for 60 biljon SEK for 4.2 miljon Swedes
- Prototyping development method
- Platform: Oracle 8, Linux, Perl, NT, VB, ASP
- Reservsystem for the mainsystem, a "garageproject" with limited req. specifications
- Restriction to make it simple but safe

CAP GEMINI ERNST & YOUNG

My personal test tool experience

- SW bugs needs to be detected, managed, tracked
- Bug statistic always of interest as PM
- Not easy to choose, a large amount
- Started with SQA Manager 1995
- Have seen market leading products to selfmade tools
- General market knowledge, demonstrations, newsletters, conferences etc.

CAP GEMINI ERNST & YOUNG

Profile of the verification startup team

- Project and Test Management
- Actuarian and Excel expert
- Senior consultant, UNIX and Operation expert
- Support from PM also Excel expert

CAP GEMINI ERNST & YOUNG

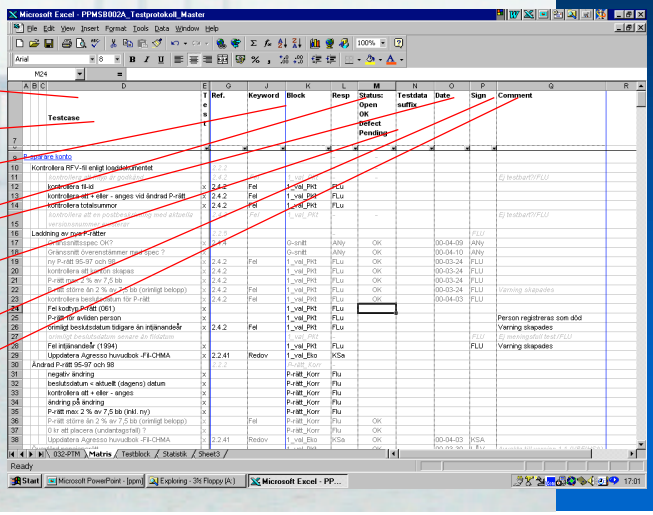
Our requirements for the verification process

- Document What to test => Test Spec's
- Define expected result
- Create Test data
- Track the test progress
- Test reporting
- Track Defect Reports and Change Requests
- Automate test execution

CAP GEMINI ERNST & YOUNG

Test specification/Protocol

Test case ID
Description
Keyword
Tester Sign
Date
Status
Tested Sign
Comment



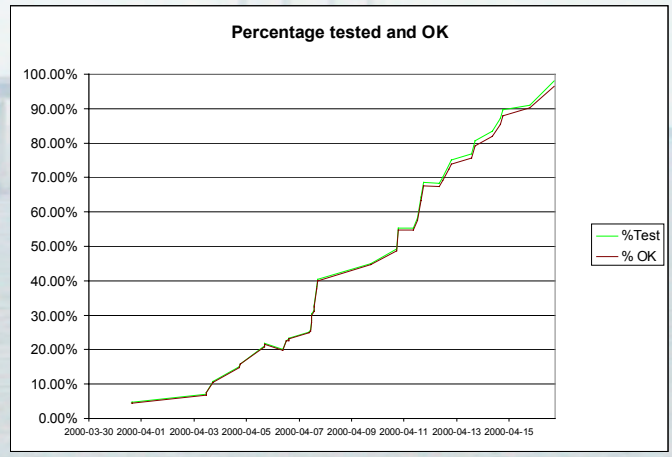
CAP GEMINI ERNST & YOUNG

Test resp. statistics

	Open	Defect	Pending	OK	
LBä	1	0	0	21	95.5%
YJo	0	0	0	0	
FLu	1	0	0	87	98.9%
SFj	0	3	0	84	100.0%
ANy	0	2	0	56	100.0%
KSa	0	0	0	30	100.0%
FAn	0	1	0	47	100.0%
KSa	0	0	0	30	100.0%
LÅV	0	0	0	7	100.0%
MBj	0	0	0	13	100.0%
Totalt	6	7	0	380	98.4%

CAP GEMINI ERNST & YOUNG

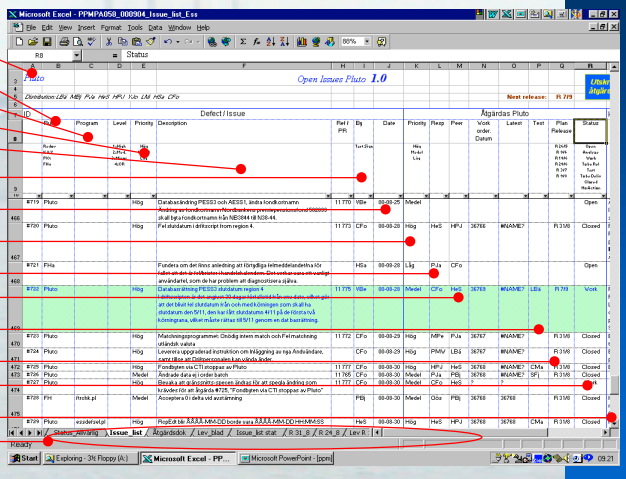
Test progress



CAP GEMINI ERNST & YOUNG

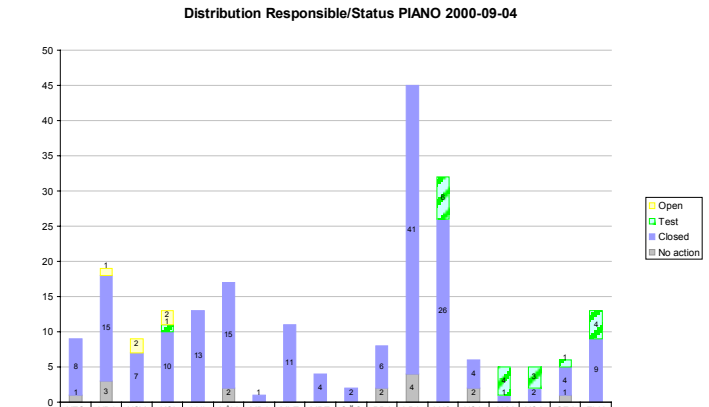
Issue list, (Defects, CR's, To Do List)

ID #
Subsystem
Program
Severity level
Description
Written by
Date
Priority
Responsible
Peer
Tester
Planned Rel.
Status
Comment
Sheets



CAP GEMINI ERNST & YOUNG

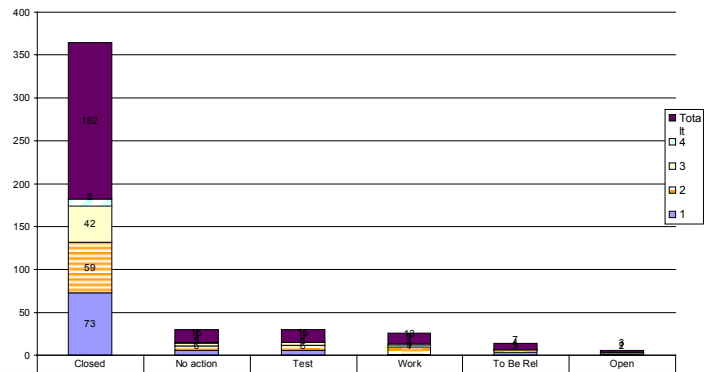
Defect Report distribution



	HES	HPJ	HSY	KSI	LNI	LÄV	MBJ	MLE	MPE	OÖS	PBJ	LBA	YJO	HSA	ANY	KSA	SFJ	FLU
Open		1	2	2														
Test				1									6		4	3	1	4
Closed	8	15	7	10	13	15	1	11	4	2	6	41	26	4	1	2	4	9
No action	1	3				2					2	4		2			1	

CAP GEMINI ERNST & YOUNG

Defect Report Status/Severity

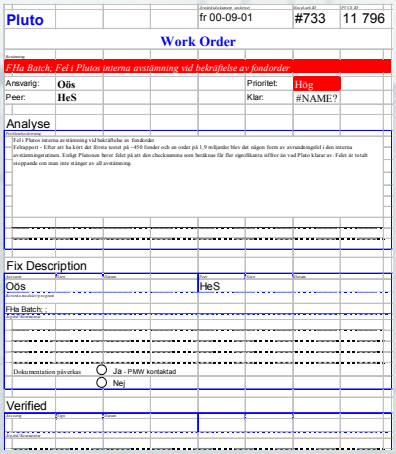


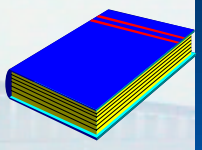
	Closed	No action	Test	Work	To Be Rel	Open
Totalt	182	15	15	13	7	3
4	8	1		2		1
3	42	3	3	1		
2	59	5	6	9	4	
1	73	6	6	1	3	2

CAP GEMINI ERNST & YOUNG

Press a button and print a Work Order

All data fetched from Issue list



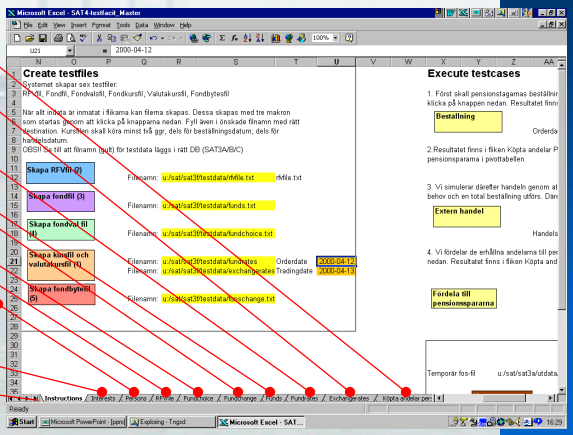


Delivery Binder
For each delivery
- Del Doc's
- Workorders
- Checklist
- Diff lists (current-previous delivery)

CAP GEMINI ERNST & YOUNG

Excel macros creates Test Data files 1(2)

Shares p-holders
Exchangerates
Fundrates
Funds
Fundchange
Fundchoice
Pensionrights
Person information
Interests
Instructions



CAP GEMINI ERNST & YOUNG

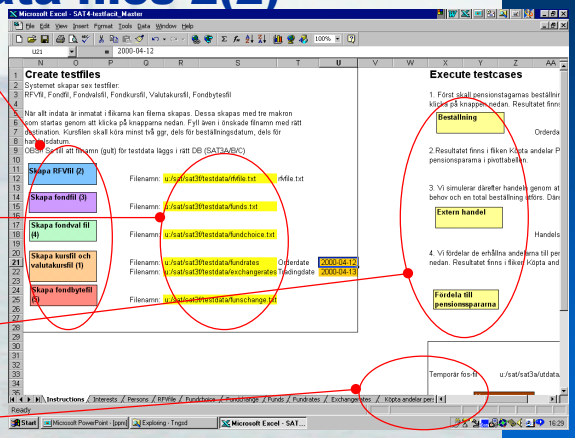
Excel macros creates Test Data files 2(2)

-Buttons for creation of test data files

- File location and name

- Buttons for calculations

- Expected result)



CAP GEMINI ERNST & YOUNG

Used Excel features

- Hyperlink to screen dumps and specifications
- Filter, Comments
- Macros in Visual Basic
  - calculations, generation of testdata, reports
- Excel Functions
- Links to Oracle DB via MS Access
- Pivot tables
- Graphs

CAP GEMINI ERNST & YOUNG

Plus factors for our test tool choice

- Small project. From 6 to 20 totally
- Office landscape
- Short distances
- Unclear Requirements
- The competence of the Verification Team
- Laptop homework analyse and editing

CAP GEMINI ERNST & YOUNG

Limitations

- Restricted write access
- Not for distributed projects
- Prejudices that a "on the market" tool is more reliable
- You have nothing to start with. (DB, Reports, defaults values, tool support from vendor)

CAP GEMINI ERNST & YOUNG

Lessons learned 1(2)

- Excel is simple but has advanced capabilities
- Nice feeling. "I decide the design content and usage, not the opposite"
- Easy to reuse the tool itself, or ideas to other projects or your next project
- Copy and paste to Word useful
- Powerful links to MS Office tools and DB

CAP GEMINI ERNST & YOUNG

Lessons learned 2(2)

- Easy to tailor to changed project objectives
- Evolutionary development of test tools is the best method
- Always available, PVCS Tracker using Oracle DB in neighbour project were not
- Start up day one.

CAP GEMINI ERNST & YOUNG

My recommendation

- "Check the capability and possibilities with Excel towards your needs and requirements before introducing any other Test Administration tool"

CAP GEMINI ERNST & YOUNG

QWE2000 Extra Session

Mr. Richard Kasperowski (Altisimo Computing)

Opportunistic Software Quality

Key Points

- Opportunistic software testing: a strategy for improving quality with limited resources
- Examples from a recent project
- Data and results

Presentation Abstract

I recently worked on a project for which I was the sole QA person and the ship date was only three months away. The project was to build and deliver a web-based billing system for the local telephone customers of a large telephone company. The team used HTML and JavaScript to present bills, a relational database at the back end, and Java and Enterprise Java Beans in the middle.

The development team consisted mainly of highly skilled designers and programmers who knew little about quality assurance. They were motivated to deliver a great product on time. My job was to help them do that, by inventing and executing a QA program for the project.

I was the only team member concerned primarily with product quality, and the delivery date was firm. I didn't have time to develop a well thought-out plan to assure the goodness of the product. Instead, I kept my eyes open for opportunities to improve product quality, and took advantage of these opportunities. I later realized that I have done this many times over the years, and began to think of my capitalizing on quality improvement opportunities as a general strategy--"opportunistic software quality."

Here are examples of the opportunities we discovered during this project and how they helped us deliver a high quality product on time:

- Configuration management: Prior to my joining, the team was already using a configuration management system. They used the system to track source code changes and to label the configurations that were delivered to their customer. I increased the rigorous use of this system, making it easier to identify the configuration that corresponded to the one the customer was using, and thus making it easier to fix the defects the customer identified.
- Bug tracking system: Again before I joined, the team put a bug tracking

system in place. They used it sporadically and didn't record all defects in it. I became the manager of the bug tracking system, making sure all defects were recorded and addressed. We didn't forget to fix any of the defects we needed to fix, so the product we delivered was better than otherwise.

- Automated nightly builds: Infrequent source code builds are a quality problem. "Code rot" can occur: individual programmers' code changes can break the compilation of another programmers' code. Broken builds can be difficult to repair if the defect was introduced too long ago for the programmer to remember why he changed the code. On previous projects, I found that it is possible to find defects simply by compiling the source code and building the product at regular intervals. On this project, I established a system for automatically building the product every night, after the programmers had gone home. The programmers agreed that they would only check-in source code changes that would at least compile correctly. We found a number of defects with this method. Because we found the defects the within a day of the code change, it was easier for the programmers to fix the defects than it would have been if we hadn't built the code regularly. In addition, builds usually succeeded when they were needed most, such as for an emergency patch release.
- Automated nightly testing: The programmers had built a number of semi-automated tests for particular sections of code. I fully automated the existing tests. Over time, I added tests of other important sections of code. I built a system for automatically executing the tests and analyzing the results. I augmented the nightly builds with the nightly automated tests. This also helped us find new defects within a day of a broken code change, making it easier for the programmers to fix the defects than it would have been if we hadn't built the code regularly.
- QA web site: The team's philosophy was that delivering a high quality product was a group effort, and not merely my responsibility. To give the other team members a view of the state of the product's quality, I built a QA web site for the team. The web site consisted of the most recent automated test results, a way to compare any set of test results with any other set of test results, hyperlinks to web pages that could be used for manual testing, and hyperlinks to written procedures. The test results section of the site made it easy for me to analyze nightly results and update baseline results. The hyperlinks to web pages that could be used for manual testing turned out to be extremely useful for the other team members; they vociferously complained whenever the site was down.
- Manual testing: It wasn't practical to automate all testing, especially GUI testing. I adapted an old test script so we could use it to test the current version of the product. I made it a policy that the script had to be executed at least once per week, and I rigorously followed the policy. With this regular planned testing, augmented by ad hoc testing from other team members, we identified many defects.
- Source code compiler: One of the team members recommended a better

Java source code compiler. I modified the nightly build-and-test scripts to use the new compiler. The new compiler used a stricter definition of the Java language; we identified a few defects simply by using the new compiler.

- Improved delivery: One of my responsibilities was to deliver patch releases and beta releases to the customer. Before I joined the team, these deliveries were time consuming and plagued with mistakes. I developed a set of scripts to automatically build and install the product, and a procedure for making the delivery, drastically reducing the number of delivery mistakes.
- HTML validation: On a previous project, I used a tool to automatically check the validity of the HTML code the product delivered to users' web browsers. I used the tool on this project and successfully found a few defects.

There were a few techniques we considered using, but did not use. These were: automated GUI testing, code coverage analysis, exhaustively testing all source code, retrofitting pre-existing tests to a better test framework, and reorganizing the source code to improve building and maintenance.

Our results were good. The customer discovered only 7% of the total number of known defects. The two best techniques for identifying defects were manual testing and automated nightly testing. Manual testing was responsible for finding 46% of known defects, and automated nightly testing helped us find 29% of known defects. (These figures are slightly out of date; I will present current figures in the final version of the paper and at the conference.)

We delivered a relatively high quality product on time. The customer accepted the delivery and found very few defects. The opportunistic strategy worked.

Author Bio...

Richard Kasperowski is president of Altisimo Computing, a software development consulting firm based in Cambridge, Massachusetts. Richard has worked as tester, developer, manager, and consultant since 1988. He has a degree from Harvard University, is a member of the ACM, and usually cycles to his clients' offices.

Opportunistic Software Quality

Richard Kasperowski
mailto:kasper@altisimo.com

Altisimo Computing
http://www.altisimo.com

Altisimo

Outline

- Introduction
- Opportunistic software quality
  - Definition
  - Examples
- Results

Altisimo

Introduction

- The project
  - Web- based billing system
  - HTML, JavaScript, RDBMS, Java, EJB
  - Deadline 3 months away
- My role
  - Sole QA person
  - Develop & enact QA program

Altisimo

Opportunistic Software Quality

- Strategy:
  - Identify quality improvement opportunities
  - Capitalize on them
- Identify: anything goes, as long as there's pay- back
- Capitalize: effective, persistent, reusable

Altisimo

Configuration management

- *Problem:* Need configuration management system
- *Opportunistic solution:*
  - CVS was in use during previous phase; continued use for current phase.
  - More rigorous use of labels
- *Results:*
  - CVS is good enough
  - Easier maintenance

Altisimo

Defect tracking system

- *Problem:* Need defect tracking system
- *Opportunistic solution:* JitterBug
- *Results:*
  - JitterBug is good enough
  - Not as good as low- end commercial systems

Altisimo

Automated nightly builds

- *Problem:* Source code rots if you build it infrequently.
- *Opportunistic solution:*
  - Scripts to build system unassisted
  - cron to automate and build nightly
- *Results:* Found new defects within 1 day of injection

Altisimo

Automated nightly testing

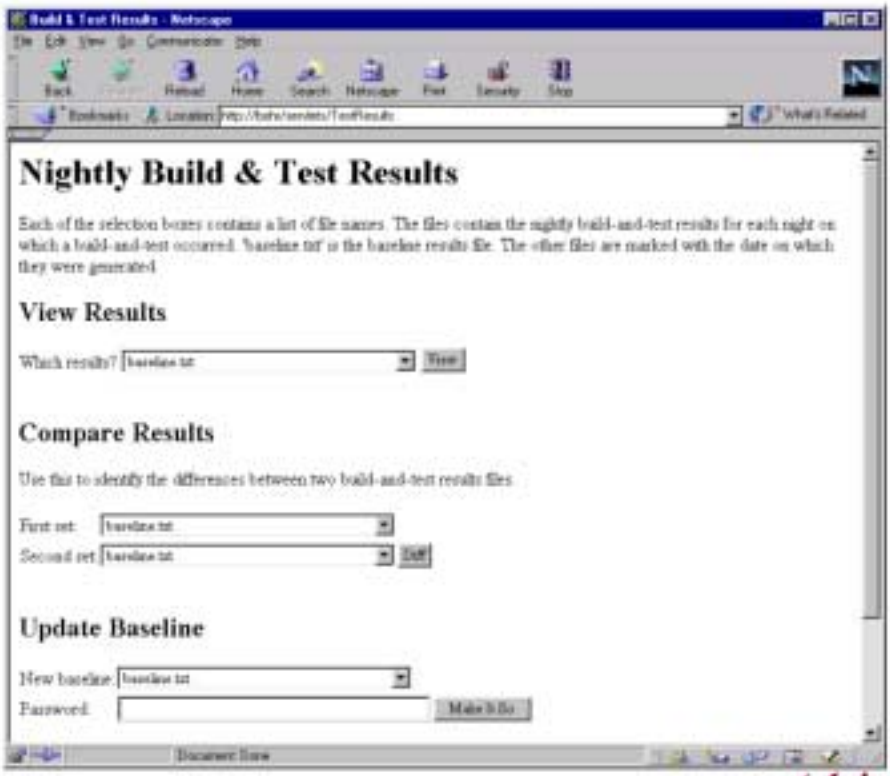
- *Problem:* Too many semi- automated tests to be able to execute them all regularly
- *Opportunistic solution:*
  - Wrapper script executes tests, savings results in log
  - diff & sed to compare results logs with baseline
  - Nightly build → nightly build- and- test
- *Results:* Found new defects within 1 day of injection

Altisimo

QA web site

- *Problem:*
  - Nightly logs too large to receive as email
  - Other team members need view of state of quality
- *Opportunistic solution:* QA web site
- *Results:*
  - Easy to compare results with baseline
  - Mixed results on use by other team members

Altisimo



Altisimo

Manual testing

- *Problem:* Impractical to automate GUI testing
- *Opportunistic solution:*
  - Reuse leftover test script
  - Execute weekly
- *Results:*
  - Found most defects
  - Successful because all team members were able to participate

Altisimo

Source code compiler

- *Problem:*
  - Conformance to Java Language Specification
  - Compile time
  - Cross- platform portability of build scripts
- *Opportunistic solution:* jikes
- *Results:*
  - Identified few new defects by itself
  - But decreased maintenance

Altisimo

Improved delivery

- *Problem:* Ad hoc delivery system introduced defects
- *Opportunistic solution:*
  - Carefully written procedures
  - Automation
- *Results:* No defects injected by delivery

Altisimo

HTML validation

- *Problem:* Can't test generated HTML on every combination of browser, OS, and bill data
- *Opportunistic solution:*
  - Custom web spider
  - weblint
- *Results:* Discovered surprisingly few HTML defects

Altisimo

Opportunities not taken

- Automated GUI testing
- Code coverage analysis
- Develop new tests
- Adapt old tests to test framework
- Reorganize Java source code tree

Altisimo

Results

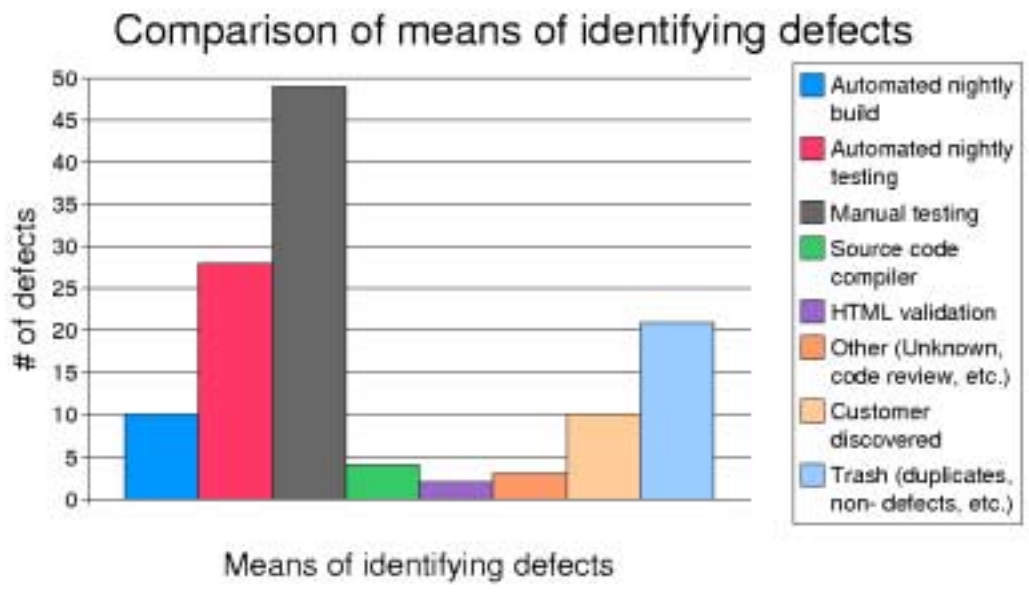
- Identified 106 unique new defects
  - 127 potential new defects identified
  - 21 duplicates, non- defect issues
- 10 post- delivery defects
- Manual testing most successful: 49 defects
- Automated nightly build & test next most successful: 38 defects

Altisimo

Results

Means of identifying defects	Number of defects	% of total	% of non-trash
Automated nightly build	10	7.9%	9.4%
Automated nightly testing	28	22.0%	26.4%
Manual testing	49	38.6%	46.2%
Source code compiler	4	3.1%	3.8%
HTML validation	2	1.6%	1.9%
Other (Unknown, code review, etc.)	3	2.4%	2.8%
Customer discovered	10	7.9%	9.4%
Trash (duplicates, non-defects, etc.)	21	16.5%	
Totals	**127**	**100.0%**	
Totals, non-trash	**106**	**83.5%**	**100.0%**

Altisimo



Altisimo

Conclusion

- Team delivered high quality product on time.
- Customer was satisfied.
- Opportunistic strategy worked.

Altisimo

Opportunistic Software Quality

Richard Kasperowski

Altisimo Computing
28 Regent Street
Cambridge, Massachusetts 02140
http://www.altisimo.com/

Email: kasper@altisimo.com

Abstract: *I recently acted as the sole quality advocate on a software project that had a rapid development schedule. As the project progressed, I realized that I was addressing the need for quick and productive quality assurance activities in a way that was similar to what I had done on other projects—I was identifying and acting on the best opportunities for quality improvement, iteratively improving the product's quality as I went. I call this strategy "opportunistic software quality."*

In this paper I present this strategy, using examples from the project described above. For each of the opportunities I identified, I present the number of software defects the opportunity helped me find. For this project and others on which I have worked, this strategy for improving software quality was successful.

Keywords: *Rapid software testing, techniques for quality assurance and quality improvement, short product development schedule, opportunistically improving software quality*

1. Introduction

I recently worked on a project for which I was the sole QA person and the ship date was only three months away. The goal of the project was to build and deliver a web−based billing system for the local telephone customers of a large telephone company. The software development team used HTML, JavaScript, and Java servlets to present bills, a relational database at the back end to hold customer and bill information, and Java and Enterprise Java Beans (EJB) in the middle for business logic.

The development team consisted mainly of highly skilled designers and programmers who had little experience with quality assurance. They were motivated to deliver a great product on time. My job was to help them do that, by inventing and executing a QA program for the project.

I was the only team member concerned primarily with product quality, and the delivery date was firm. I didn't have time to develop a well thought−out plan to assure the goodness of the product. Instead, I continuously looked for opportunities to improve product quality and took

---

Richard Kasperowski is president of Altisimo Computing, a software development consulting firm based in Cambridge, Massachusetts.

advantage of these opportunities. I later realized that I have done this on many projects over the years, and began to think of my capitalizing on quality improvement opportunities as a general strategy—"opportunistic software quality."

2. Opportunistic software quality

The strategy is to identify opportunities to improve software quality and to take advantage of these opportunities. To take advantage of the opportunities is to turn them into effective, persistent, reusable processes and procedures. Over the life of a project, one−time hacks are not as useful as repeatable, sustainable quality improvement activities. The value of this strategy lies in the improvement to the product gained by the repeated application of the opportunities.

In the following sections, I present examples of the opportunities we identified during this project and how they helped us deliver a high quality product on time.

2.1 Configuration management

Problem: Configuration management (CM) is an important part of any large project. Most CM systems share a few characteristics. They make it possible to track revisions of individual files, including who made the change, why the programmer made the change, and the content of the change. By branching source code into multiple trees, CM systems make it possible to maintain a production version of a software system on one branch while enhancing the system on an independent branch. CM systems also let you label a particular set of files and their revision numbers, making it easy to identify exactly which set of files corresponds to a particular release or bug fix.

Opportunistic solution: Prior to my joining, the team was already using CVS [SourceGear] as the project's configuration management system. They used CVS to track revisions to individual files and to label major releases using CVS's tag feature. I increased the use of CVS by creating branches for maintenance and for further development. I also began labeling each patch release, making it possible to identify the configuration that corresponded to the one the customer was using in production, thus making it easier to fix the defects the customer identified.

Results: Our overall experience with CVS was that it is a reasonable CM tool. It does a fine job tracking revisions to individual files and enabling the identification of sets of files and their revisions. Unlike some more advanced CM tools, CVS doesn't directly facilitate associating code changes with defect IDs; we worked around this deficiency by keeping a separate database of release labels and defect IDs.

CVS is a little different from other CM systems in that you don't have to exclusively lock a file in order to edit it. Because the files a programmer is working on aren't marked as "locked," programmers sometimes forget to check in all their changed files to the central repository; the result is that a programmer might fix a defect in his private "sandbox" but not fully propagate the fix to the central repository.

CVS isn't for programmers who need a graphical front−end on their CM system. Graphical user interfaces (GUIs) are available for CVS, but they do not fully support all the CVS features available on the command line. Using CVS via the command line is thus more powerful than using it via a GUI, but with command line use it can be easy to make mistakes.

2.2 Defect tracking system

Problem: A defect tracking system is another important tool on large projects. Without a defect tracking system, it is easy to forget to fix defects you know exist. Another problem is that one team member can be unaware that another team member is actively solving a problem; the two team members might duplicate each other's efforts. "Number of defects" is a basic quality metric, but it is difficult to measure without a defect tracking system. Finally, without a defect tracking system there is no specific history of what went wrong (the defects injected) while developing the system and how the team addressed the defects.

Opportunistic solution: Again before I joined, the team put JitterBug [Tridgell] in place as the project's bug tracking system. JitterBug is a free defect tracking system used on open source software projects such as `jikes` [IBM]. The team used JitterBug sporadically and didn't record all defects in it. I became the manager of the bug tracking system, making sure all defects were recorded and addressed. We didn't forget to fix any of the defects we needed to repair, so the product we delivered was better than it otherwise would have been.

Results: JitterBug is not as good as low−end commercial defect tracking systems, but it served our needs nonetheless. By the end of the project, we identified and recorded 177 defects in JitterBug, including defects identified during the first phase of the project, duplicates, and issues that turned out not to be defects. The customer perceived a high level of quality in the product we delivered; the perceived level of quality undoubtedly would have been lower if we had not been so careful about tracking and fixing defects.

2.3 Automated nightly builds

Problem: Infrequent source code builds are a quality problem. "Code rot" can occur: when old code is not compiled frequently, new or modified code can make the old code uncompilable. Broken builds can be difficult to repair if the code changes were introduced too long ago for the programmer to remember how he changed the code.

Opportunistic solution: An easy way to assure quality is to appoint a build master who makes sure he can build the code, with no compiler errors, at regular intervals. The interval we chose was one day. Programmers agreed that by the end of each day, any code changes they had checked in to the CVS repository would build correctly.

The best way to build the system regularly is to do it automatically. Before I joined the team, they used a set of build scripts to compile all the source code and build EJB components. The scripts were interactive, prompting the user for which components he wanted to build. As build master, I automated the build scripts, replacing interactive prompts with command line arguments. I added a wrapper script that would backup the previous instance of the source tree, check out the latest source code, and build the whole system. I used `cron` to run the wrapper script automatically each weekday night at 2:00 AM. The wrapper script emailed a log of the build to me; each morning I reviewed the log for build errors, isolated the source of each error, and asked the programmer who injected the error to fix it. This system was known as the "nightly build."

Results: We found a number of defects with this method. The usual problem was that the programmer didn't check in all of his changes, so some source files in the repository were inconsistent with each other. Because we found the defects within a day of the code change, it was easier for the programmers to fix the defects than it would have been if we hadn't built the

code regularly. In addition, builds usually succeeded when they were needed most, such as for an emergency patch release.

2.4 Automated nightly testing

Problem: The programmers had built a number of semi−automated tests for particular sections of code. We wanted to run the tests regularly to help us identify newly introduced defects, but there were too many tests to be able to run them manually.

Opportunistic solution: I initially attempted to automate the existing tests by adapting them to the Java Test Driver [Kasperowski]. Because the tests were not designed for that kind of test framework, however, it was time consuming, and I soon realized that I wouldn't be able to finish porting the tests before the ship date. Instead, I built a wrapper script to simply execute the existing tests in sequence. I added this script to the nightly build, which became the "nightly build−and−test."

To analyze the results of each nightly build−and−test, I built a means of automatically comparing the current results log with the results log of any other run of the nightly build−and−test. My first attempt to do this was to use `diff` for a simple file comparison. There were two problems with this approach: (1) the application's running log contained time stamps, which were expected to be different every day, and (2) the application's running log contained verbose Java garbage collector messages, which by their nature are nondeterministic. I minimized (but did not completely solve) this problem by augmenting the simple `diff` with a number of `sed` filters to produce a reasonable build−and−test log comparison tool.

Over time, I added tests of other important sections of code, using the Java Test Driver as the test framework for the new tests.

Results: Automated nightly testing helped us find new defects within a day of a broken code change, making it easier for the programmers to fix the defects than it would have been if we had not built the code regularly.

2.5 QA web site

Problem: The nightly build−and−test logs were each over one megabyte in size; I grew tired of receiving these huge logs as email attachments. I also didn't want to have to remember the long command line required for comparing two sets of nightly build−and−test logs. Finally, I was the only team member with a view of the results of the nightly build−and−test; I wanted the whole team to have a view of the quality of each night's build−and−test.

Opportunistic solution: I built a QA web site for the team, using the Apache web server [Apache], the Apache JServ servlet engine [JServ], and Java servlets [Davidson]. The web site, shown in Figure 1, consisted of the most recent automated test results, a way to compare any set of test results with any other set of test results, hyperlinks to web pages that could be used for manual testing, and hyperlinks to written procedures.

Results: Using servlets to build a web site was relatively easy, but it was difficult to learn how to configure JServ. (My motivation for implementing this web site with servlets was that the project used servlets in its implementation, and I wanted to understand how they worked.) The web site made it very easy for me to review nightly build−and−test results.

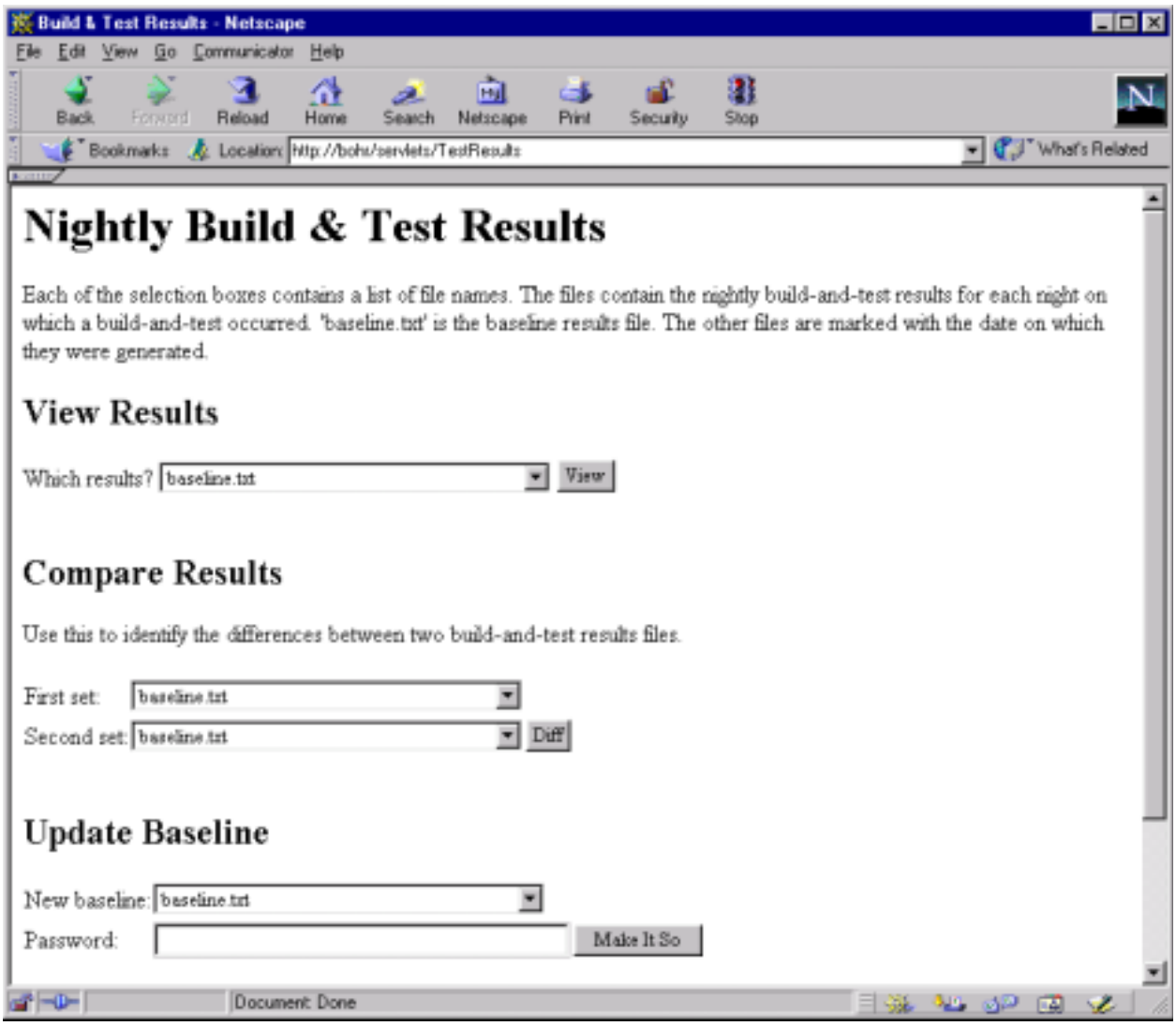


Figure 1: The nightly build−and−test web site

The test results web site grew into a more general project web site, with hyperlinks to sample test bills and deployment procedures. The hyperlinks to web pages that could be used for manual testing turned out to be extremely useful for the other team members; they vociferously complained whenever the site was down.

Unfortunately, few team members used the test results part of the web site. They were more interested completing the implementation than in viewing the day−to−day results of the nightly build−and−test.

2.6 Manual testing

Problem: We wanted to evaluate the quality of the GUI regularly, but it wasn't practical to automate GUI testing. One reason was that the team had not acquired automatic GUI testing tools and it would have taken too long to purchase one. Another reason was that the GUI design was not frozen, so it would have been a poor use of our time to develop and maintain automated GUI tests.

Opportunistic solution: A written script for testing the GUI was left over from the first phase of

the project, and I adapted it for use with the current version of the product. I made it a policy that the script had to be executed at least once per week, and I rigorously followed the policy.

Results: With this regular planned testing, augmented by ad hoc testing from other team members, we identified many defects. In fact, manual GUI testing identified more defects than any other part of the quality program. The success of the manual GUI testing effort probably stems from the fact that all team members participated; the other quality initiatives were executed by me alone.

2.7 Source code compiler

Problem: Different Java compilers have different characteristics. One important characteristic is whether the source code they accept conforms to the Java Language Specification [Gosling]. If a compiler accepts non−conforming source code, the compiled code's behavior might be unpredictable. In addition, non−conforming source code is more time−consuming to maintain than conforming source code.

Another important characteristic of a Java compiler is the amount of time it takes to compile the source tree. Programmers are unlikely to use the project's build scripts to test whether their code is buildable if it takes too long to execute a whole−system build.

A third characteristic of a Java compiler is its cross−platform portability. The usual Java compiler, Sun's `javac` [Sun], is portable in that it is written in Java, but the format of the path names in its command line parameters varies depending on whether it is running on Windows or UNIX. Because of this, we were maintaining two sets of build scripts, one for Windows and one for UNIX. As the build scripts evolved, it was difficult to keep the Windows−specific build scripts synchronized with the UNIX−specific ones. Programmers sometimes complained that the Windows−specific build scripts were broken; the scripts were indeed broken, because we often forgot to update them to match the behavior of the UNIX−specific build scripts.

Opportunistic solution: One of the team members suggested that we use `jikes` [IBM] as our Java compiler. `jikes` rejects non−conforming source code. `jikes` is written in C++ and is compiled into a platform−native executable, so its execution speed is faster than that of `javac`. `jikes` accepts UNIX−style path names (that is, paths with forward slashes) on both Windows and UNIX. I modified the nightly build−and−test scripts to use `jikes` by default.

Results: The amount of time it took to build the whole system decreased from 63 seconds to 16 seconds, an improvement of nearly four times. (Times were measured on a Sun Enterprise 250 with 2 CPUs, 512M bytes of RAM, and 27G bytes of total disk space.)

Because `jikes` accepts UNIX−style path names on both Windows and UNIX, I was able to retire the Windows−specific build scripts. I no longer had to maintain two sets of build scripts, struggling to keep them synchronized. Programmers were more likely to use the build scripts because the single set of scripts was more likely to succeed.

We identified a few instances of non−conforming Java source code. Two kinds of errors were typical: unreachable code and uncatchable exceptions.

2.8 Improved delivery

Problem: Delivering the product to the customer and installing it on the customer's machine was an ad hoc procedure. It was difficult to repeat successful instances of deployment. The

installation guide was too difficult to follow, so the person installing the system usually made mistakes. It was time consuming to debug each unsuccessful deployment.

Opportunistic solution: I developed a set of scripts to automatically build and install the product, as well as a procedure for making the delivery; this drastically reduced the number of deployment mistakes. I began by carefully recording the steps I followed to build, deliver, and install the system. I used this list of steps as the de facto procedure for deployment, refining it every time I deployed the system. When I was comfortable deploying the system manually, I built a script to do the work for me. Thereafter, the deployment procedure consisted of my following a few manual steps and running the script.

Results: Deployment was almost always successful. There were no defects injected by poor deployment.

2.9 HTML validation

Problem: The product is web−based, but there weren't enough resources to be able to test the GUI on every combination of web browser and operating system. How could we be confident that the application would generate valid web pages for arbitrary combinations of web browser and operating system, and for arbitrary customer billing information?

Opportunistic solution: On a previous project, I discovered a tool called `weblint` [Bowers] to automatically check the validity of the HTML code that the product delivered to users' web browsers. `weblint` validates HTML files against the World Wide Web Consortium's HTML 3.2 standard [W3C]. `weblint` also has parameters to validate the HTML extensions accepted by Microsoft's Internet Explorer and Netscape's Navigator browsers.

The billing application generated web pages dynamically based on the user's monthly account information and on what kind of page he requested. `weblint` can only validate static HTML files, though. One of the team members had built a simple web spider that visited parts of a customer's bill to ensure they exist. I enhanced the spider to visit every page of a customer's bill, saving the dynamically generated HTML files to disk. I created a large number of representative bills, saved their HTML to disk, and ran `weblint` on the disk files to validate the generated HTML. I added the execution of the web spider and `weblint` to the nightly build−and−test.

Results: The generated HTML was surprisingly good—we identified only a small number of defects this way. I write "surprisingly good" based on my experience with other web testing projects, where the HTML was largely non−conforming with respect to the standard.

When I ran `weblint` in Netscape−compatibility mode, it identified many instances of non−compliant HTML code. These were not considered defects because the web pages were designed to use Internet Explorer HTML extensions.

2.10 Opportunities not taken

There were a few techniques we considered using, but did not use. They were good ideas and might have identified defects, but we did not have the resources to execute them.

One of our ideas was to automate GUI testing. Unfortunately, we didn't already have a GUI testing tool in our possession, and we probably didn't have enough time to develop and maintain automated GUI tests anyway. In addition, the GUI was in a state of flux, so adjusting the tests to

match a given day's instance of the GUI would have been costly. "The cost of automating a test is best measured by the number of manual tests it prevents you from running and the bugs it will therefore cause you to miss;"[Marick] by this measure, automating the GUI testing would not have been a good investment.

We considered performing code coverage analysis to help determine the goodness of our existing tests and to guide new test development. We went so far as to evaluate tools, select one, and place an order for it. In this case, big−company bureaucracy impeded us. Two months after my placing the order, the tool had not arrived; the project was nearing its end date, and we would not have enough time left to use the tool effectively. In retrospect, this delay might have been a good thing: we had plenty of other development work to do, and the use of the code coverage tool might have interfered with other quality assurance activities.

We did not build many new tests. I began building a tool that would identify dependencies between Java classes. I wanted to test the classes exhaustively, in order from those with the least number of dependencies to those with the greatest number of dependencies, using a levelization technique similar to the one described in [Lakos]. However, while building this tool, I would not have been identifying defects, so I dropped the idea. We were fairly comfortable with the existing code anyway, despite that it had not been extensively tested in the lab. Much of the code was already running in production and working well in that environment.

The semi−automated tests that existed before I joined the team exercised large sections of code and sent the results to stdout. I wanted to modify these tests so they would exercise smaller sections of code, to make it easier to identify the source of a defect when one was found. I also wanted the tests to be able to determine automatically whether their results matched the expected results, and report that information to the human tester. I began retrofitting the existing tests to work within the Java Test Driver framework. This proved to be too time consuming; I estimated that I would have spent all my time on this activity and nothing else.

Finally, we wanted to reorganize the Java source code tree. The existing source code tree did not follow the Java convention. The convention is that the source code tree is a tree of directories whose names match Java package names. The .java files that declare themselves to be in a given package go in the corresponding directory. Instead of following this convention, the files in the source code tree were grouped by module name, which was independent of the Java package names. This is a problem because Java compilers and other development tools do not work efficiently, if at all, unless the .java files are in a conventional Java source tree. It also makes maintenance difficult because experienced Java programmers have difficulty finding source code files if they are not in the conventional directory tree. I built a tool that takes an arbitrary Java source tree as its input and constructs a conventional Java source tree. The tool appeared to work, but we were afraid of the potential instability that reorganizing the source tree might introduce. We postponed this activity until the next phase of the project.

3. Results

We identified 127 potential defects during this phase of the project. Of these, 21 were duplicates or were issues that were not really defects, leaving 106 unique defects. The customer discovered at most 10 of these defects, or 9% of the total number of known defects. In fact, the customer discovered fewer than 10 defects: 10 is the sum of the number of defects identified by the customer and the number of defects we identified while investigating those defects. Although we did not establish a specific target at the beginning of the project, our delivering 10 defects to

the customer indicates that we were successful in delivering a high quality product.

The two best techniques for identifying defects were manual testing and automated nightly testing. Manual testing was responsible for finding 49 defects, or 46% of known defects, and automated nightly testing helped us find 28 defects, or 26% of known defects. The combination of automated nightly build and automated nightly testing together identified 38 defects, or 36% of known defects.

Table 1 and Figure 2 show the number of defects identified by each defect identification activity.

We delivered a relatively high quality product on time. The customer accepted the delivery and found very few defects. The opportunistic quality improvement strategy worked.

Means of identifying defects	Number of defects	% of total	% of non−trash
Automated nightly build	10	7.9%	9.4%
Automated nightly testing	28	22.0%	26.4%
Manual testing	49	38.6%	46.2%
Source code compiler	4	3.1%	3.8%
HTML validation	2	1.6%	1.9%
Other (Unknown, code review, etc.)	3	2.4%	2.8%
Customer discovered	10	7.9%	9.4%
Trash (duplicates, non−defects, etc.)	21	16.5%	
Totals	**127**	**100.0%**	
Totals, non−trash	**106**	**83.5%**	**100.0%**

Table 1: The usefulness of each means of identifying defects is indicated by the number of defects identified by each means. "Manual testing" includes both GUI and non−GUI manual testing. "Other" includes all ad hoc means. "Customer discovered" includes defects identified by the customer, as well as defects we identified while investigating those defects. "Trash" includes duplicates and issues that were not truly defects.

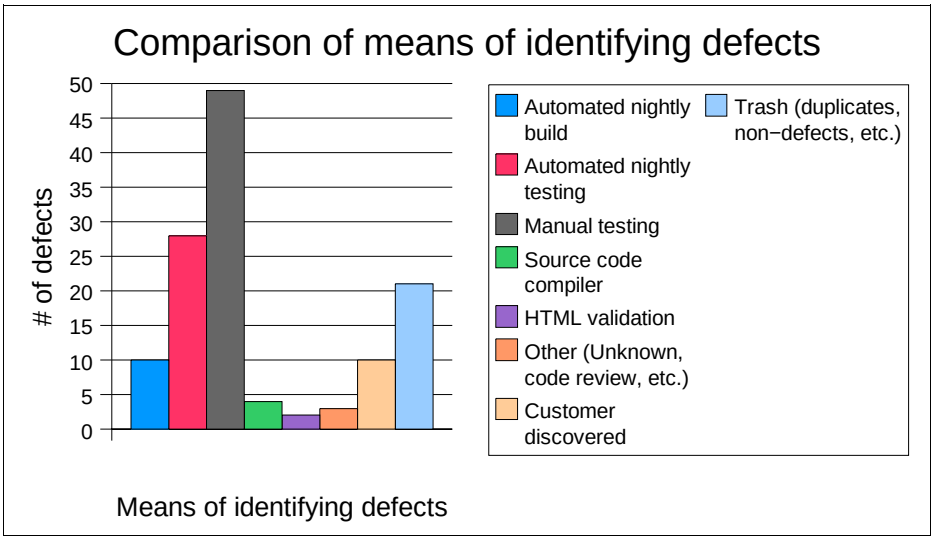


Figure 2: The relative utility of each means of identifying defects. Manual testing is clearly a productive means of identifying defects. The combination of automated nightly build and automated nightly testing is close behind.

4. References

[Apache] The Apache Software Foundation. The Apache HTTP server project. "http://www.apache.org/httpd.html".

[Bowers] Bowers, N. Weblint. "http://www.weblint.org/".

[Davidson] Davidson, J., and Ahmed, S. Java servlet API specification, version 2.1a. Sun Microsystems, Inc., 1998.

[Gosling] Gosling, J., Joy, B., and Steele, G. *The Java Language Specification*, edition 1.0. Sun Microsystems, Inc., 1996. "http://java.sun.com/docs/books/jls/html/".

[IBM] IBM. Jikes project. "http://oss.software.ibm.com/developerworks/opensource/jikes/project/".

[JServ] The Apache Software Foundation. The Apache JServ project. "http://java.apache.org/jserv/".

[Kasperowski] Kasperowski, R. The design and implementation of a Java test driver. In *Proceedings of the 16th International Conference and Exposition on Testing Computer Software (*Washington, D.C.), 1999; "http://www.altisimo.com/research/design-implement-test-driver.html".

[Lakos] Lakos, J. *Large-scale C++ Software Design*, Addison-Wesley, 1996, 203-324.

[Marick] Marick, B. When should a test be automated? In *Conference Proceedings: Eleventh International Software Quality Week* (San Francisco), May, 1998.

[SourceGear] SourceGear Corporation. CVS. "http://www.sourcegear.com/CVS/".

[Sun] Sun Microsystems, Inc. Java 2 SDK tools. "http://java.sun.com/products/jdk/1.2/docs/tooldocs/tools.html".

[Tridgell] Tridgell, A. JitterBug. "http://samba.anu.edu.au/jitterbug/".

[W3C] World Wide Web Consortium. HTML 3.2 Reference Specification. Jan., 1997. "http://www.w3.org/TR/REC-html32.html".

QWE2000 Extra Session

Mr. Olivier Denoo [Belgium]
(ps_testware)

"Assuring Your E-commerce Revenue"

Key Points

- Explaining the generic competitive strategies.
- Explanation of the Business Process Thinking technique to ensure the requirements are as desired by the customer.
- Classification of test requirements based on prioritisation on business criticality/added value and uniqueness to by able to check the chosen competitive strategy.
- Explanation of the test techniques depending on the classification.
- Comparison of intended competitive strategy on web sites and perceived competitive strategy.

Presentation Abstract

Abstract to be supplied.

About the Speaker

Olivier Denoo is a consultant active in the Business of Structured Software Testing since 1997. He was the key-developer of ps_testware's Y2K testing techniques. Nowadays, he still is a highly respected trainer of this methodology. Continuously looking for new challenges, Olivier started to investigate the possibilities of website and e-commerce testing and became in charge of ps_testware's e-commerce/WWW-testing knowledge base. Currently, he is working in a project at one of the largest Belgian Telecom companies.

QWE2000 – Assuring Your E-commerce Revenue

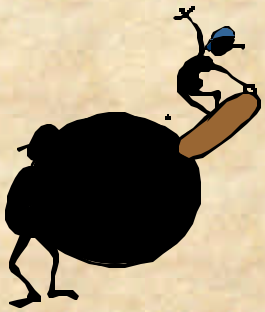
Agenda

- Introduction
- The four WebSurfers of Apocalypse
- Web benchmarking tools
- Web load / performance test tools
- Web monitoring tools
- Conclusions
- Questions

ps_testWare
Software Testing Services

Introduction

The New Economy



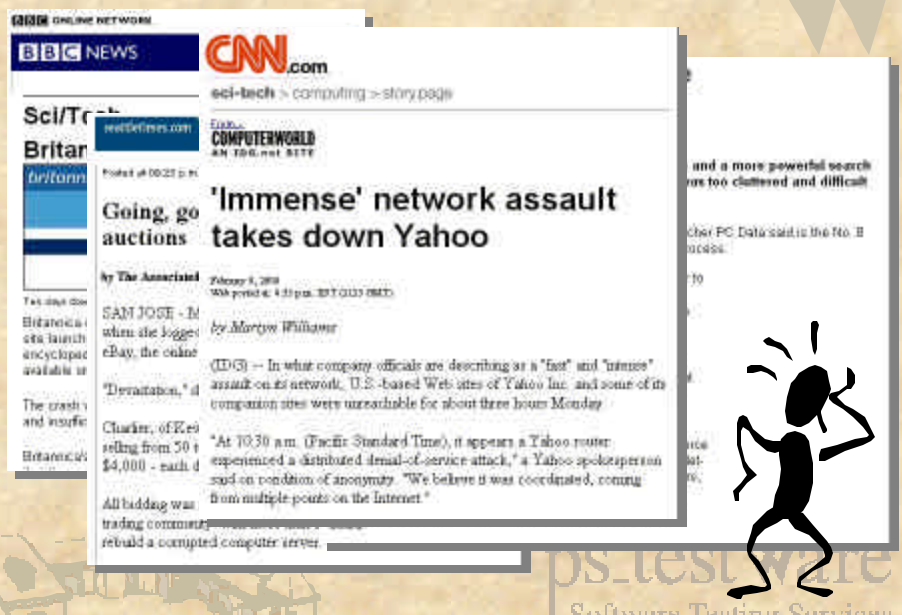
The New Economy

- **Internet is booming**
  - **200 million potential consumers are connected today**
  - **1 billion expected within 10 years**
  - **The number of E-commerce web sites is growing exponentially**

The New Economy

- **The new economy is knocking on our door**
  - **Get webbed or die**
  - **Fast-growing business**
  - **Stock exchange is surfing…its own way**
  - **Competitors are already there**

But…



The Four WebSurfers of the Apocalypse

Thanks to DC Comics & Marvel

- Availability
- Reliability
- Scaleability
- Performance

Downtime Costs:

- **Dell computers :**
  - Half of the sales are made through the web
  - Revenue is about 40 million dollars / day
- **The damages are even bigger :**
  - Lost sales
  - Lost advertising
  - Maintenance and repair costs
  - **Bad impression**
    - **Reliability**
    - **Security**
    - **./...**

Business Costs :

- **If download time is « too long »**
  - **33% just leave and do not try again**
  - **44% leave and buy the same from traditional retail sources**
  - **I would leave and buy from competitors**

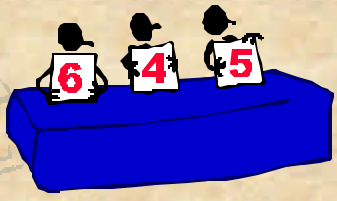
Solutions

Solutions

Simply

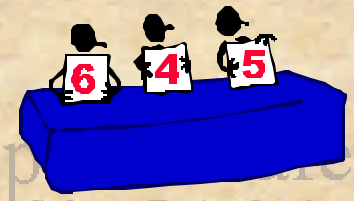
STRUCTURED TESTING

Web Benchmarking Tools



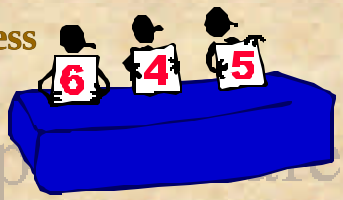
Web Benchmarking Tools

- Measure and compare performances
  - Download time
  - Accessibility
  - « Freshness » of the information
  - ./..…
- Test in production
- (Mostly) Encapsulated in services
- Cost = depending on amount and complexity



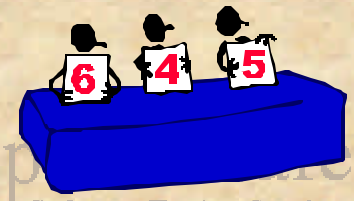
Web Benchmarking Tools

- Comparing with « competitors »
  - There' s no game to win except yours
  - Being the first is no issue, only business objectives count
- Possible confusion between benchmarking and « benchmarketing »
- Environment is not under control
  - Cache, routing, parameters, …
- Realistic = Real time
- Hard to reproduce
- Could interfere with real business
- Too late
- Cheap & flexible

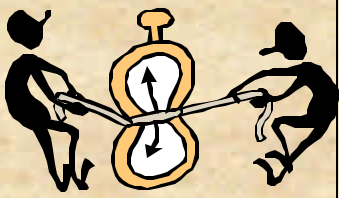


Web Benchmarking Tools

- **Some web benchmarking tools:**
  - **Holistic Performance index**
  - **WebCriteria WebCriteria Industry Index Database**

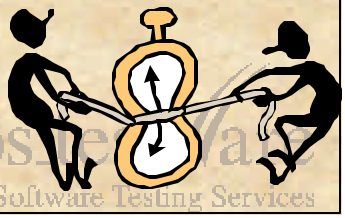


Web Load / Performance Tools



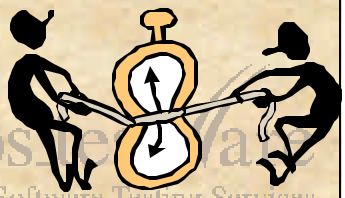
Web Load / Performance Tools

- Measure scalability
- Measure performances through simulating virtual clients (traffic simulation)
- Number of virtual users depending on computer power and software licence
- Output = nb. of users vs. Response time or any relevant metric



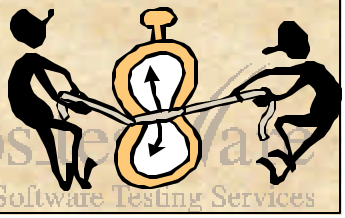
Web Load / Performance Tools

- Business as usual
  - (Re)Play normal business scenarios
  - Use combination of business profiles
  - Measure response time and performances
- Breakdown test
  - Load server until it breaks = find boundaries
    (slight increase then sudden sky high response time)
  - Define objective up front (no sadistic approach)
  - Use to validate business goals
- Cost depending on
  - Technique
  - Number of virtual users



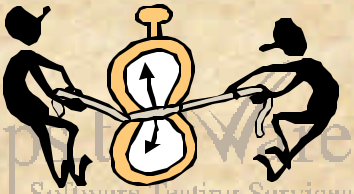
Web Load / Performance Tools

- Controllable
- Reproducible
- Predictive values
- Can fail in reproducing real world' s conditions
- Expensive (10.000 – 25.000 Euro)



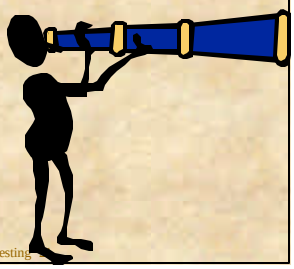
Web Load / Performance Tools

- Some web load testing tools:
  - RadView WebLoad
  - Mercury Interactive Astra Site Test
  - Web Performance Web Performance Trainer
  - Rational Performance Studio
  - Compuware QALoad
  - RSW e-Load
  - Segue Silk Performer



ps_testWare
Software Testing Services

Web Monitoring Tools



Web Monitoring Tools

- Availability
- Reliability
- Performances (download time trigger)
- Basic functionality (hyperlinks..)
- Defined scenarios repeated at different time from different time zones
- E-Alert is sent to a defined mail box
- Separate or encapsulated in services

Web Monitoring Tools

- Realistic = real time
- Helpful = quick response & warning
- Environment is not under control
  - Cache, routing, parameters, …
- Hard to reproduce
- Could interfere with real business
- Too late
- Cheap & flexible

Web Monitoring Tools

- Some web monitoring tools:
  - Compuware WebCheck
  - RSW e-Monitor
  - Segue Silkmonitor
  - Testworks CAPBAK
  - Mercury Interactive Topaz
  - Service Metrics SM-Web

Conclusions:

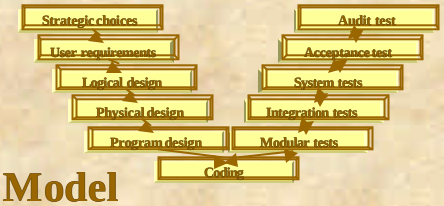
- **Which ones to use?**
  - **All = They are complementary**
- **How to include them in your testing?**
  - **Structured E-Commerce Testing / Methodology**
  - **Start testing as early as possible**
  - **Business Process Thinking ™**
  - **Understand the needs and adopt the right approach (TRH)**
  - **Plan long term (evolution, post production, maintenance..)**
- **Do not forget**
  - **Usability & Information Architecture**
  - **Functionality & Portability**
  - **Security**

ps_testWare
Software Testing Services

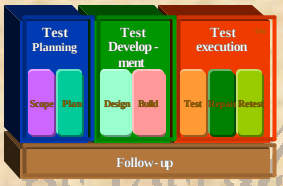
Tiensesteenweg 329
B-3010 Leuven
Tel.: +32 (16) 35.93.80
Fax: +32 (16) 35.93.88
e-mail: ps_testware@compuserve.com
http://E-Commerce.pstestware.com

Our Business

- **Structured Software Testing**
- **Methodology**



- **Implementation Model**



Our Services

- **Training** (see ps_testware institute)
- **Coaching**
- **Consultancy**
- **Outsourcing** (now also Total Outsourcing)

Provided by:

– **Test Consultants**

Our Products

- Test Assessment
- Test Assignment
- Test Plan
- Test Report
- Test Advice
- Test Pack™ New
- Test Laboratory New
- *Tools*
- *E-commerce testing*

References

- Kredietbank
- Barco Graphics
- Exact Maatwerk
- ING Bank
- Bank Card Company
- Janssen Pharmaceutica
- Tessa
- Europese Raad
- Lernout & Hauspie
- Origin
- Specs
- Dexia
- Siemens
- ING 2
- Yokogawa
- Link
- Alcatel Bell
- Mobistar
- AXA-Royale Belge

Credo

ps_testware' s first responsibility goes to the customers who use our services. Our services must be of high quality and must be a reference for our customers. In line with our primary business, Structured Software Testing, we may not indulge in pressure, quantity or quick profit.

We are responsible to our members, the men and women who work with us. Every member must be respected as an individual and must be rewarded personal and fair. We must support our members via a competent management, an adequate working environment and proper working conditions. Our members must have the means to provide and receive feedback, allow them and the organisation to learn continuously. We must support our members in their family responsibilities. Our actions must be just and ethical.

Our final responsibility is to our stockholders. Our business must make a sound profit. We must innovate and continuously improve our methods and techniques. We must develop new services and implement them effective and efficient. We must create reserves to provide for adverse times. Our stockholders must receive a fair return on their investments.

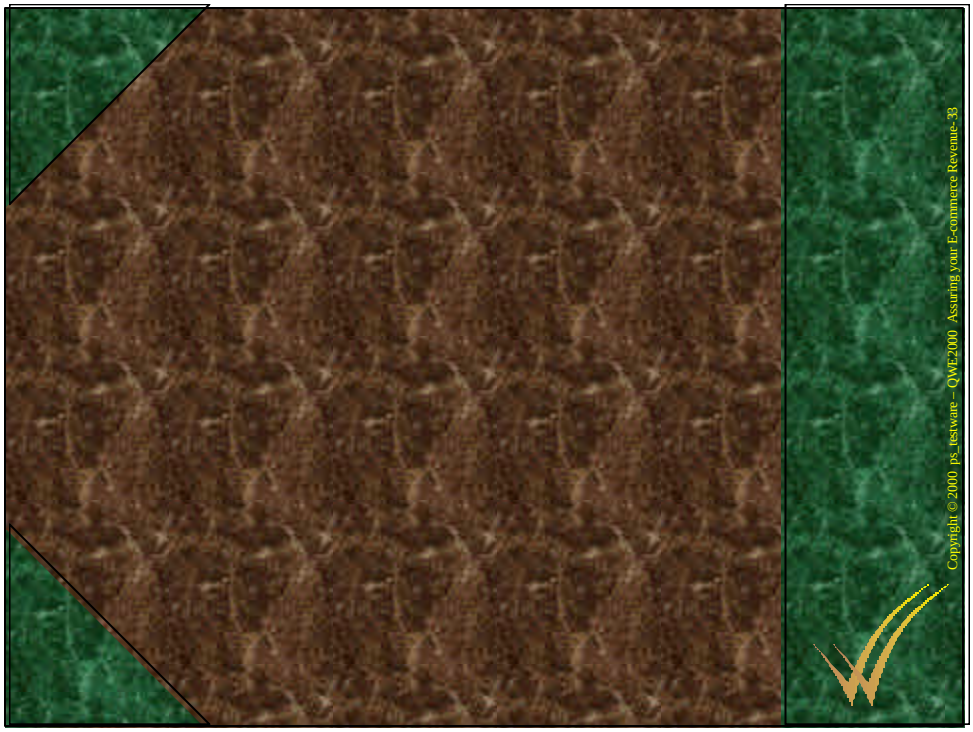
The Mission

To offer the best solution to quality problems of computer systems by using its test expert knowledge in a professional way.

***Best solution**: the solution that provides the highest contribution.*

***Test expert knowledge**: the intellectual asset of ps_testware, a profound and complete knowledge regarding verification and validation (testing).*

***Professional**: the courage to really provide what has been promised.*


Copyright © 2000 ps_testware – QWE2000 Assuring your E-commerce Revenue- 33

SAI (Studiecentrum voor Automatische Informatieverwerking) is a non-profit organization whose purpose is to promote the knowledge of Information. As far as the theoretical as practical aspects of information knowledge, the organization takes a stand on social questions that has to do with automation of information as far as it applies. The activities of the organization are focused on people who are specialists in information. SAI organizes discussions, meetings, seminars, workshops and a magazine called "Informatie". (http://www.sai.be/)

The aim of TestNet is the professionalization of testing IT products, and an increase in the awareness and importance of testing as a profession in its own right. TestNet stimulates the exchange of professional knowledge and practical experience amongst testers, and stimulates research, from a scientific standpoint as well as from a practical perspective. (http://www.testnet.org)

The Technologisch Instituut is active in all professional fields where technology and science play a central part. These include a wide range of activities, ranging from civil engineering, infrastructure, process technology, electrical engineering, telecommunications, computer technology, food and agricultural technology, the study of materials to general technologies such as management techniques, energy, safety and environmental technology (34 different sections). (http://www.kviv.be)

The American Society for Quality is a membership organization dedicated to promoting the principles and practices of quality improvement, with a mission to "be recognized throughout the world as the leading authority on, and champion for, quality." The ASQ Software Division, which represents over 5000 professionals worldwide, is dedicated to "improve the ability of individuals and organizations to satisfy their customers with quality software products and services through education, communication, research, outreach, and professional development. http://www.asq.org

The European Software Institute is one of the world's leading independent authorities on software process improvement. ESI is a non-profit-making organisation driven by the demands of European industry. It is supported by the European Commission, the Basque Government and through company membership. ESI's work is centred on products and services that are tied directly to core business objectives such as reducing costs and increasing predictability of results among others. ESI's headquarters are in Bilbao, Spain. (http://www.esi.es)

The goal of the European Systems and Software Initiative (ESSI) is to promote improvements in the software development process in industry, through the take-up of well-founded and established — but insufficiently deployed — methods and technologies, so as to achieve greater efficiency, higher quality, and greater economy. In short, the adoption of Software Best Practice. (http://www.esi.es/ESSI/welcome.html)

The German Informatics society (GI) is the association of about 21,000 men and women who, by their work, take part in the progress of informatics or who are interested in the development of informatics. The work of the GI ist mostly done in a decentralized way in divisions, expert committees, special interest groups, working groups, regional groups, advisory councils and in user groups.
(http://www.gi-ev.de)

QWE2000 Keynote Session K3-1

Rik Daems
Minister of Telecommunications
Government of Belgium

Belgium's Five-Star Plan to Develop The Information Society

Presentation Summary

The Belgian government has approved a new "Five-Star" master plan for development of the information society in Belgium. This plan covers all government departments and aims at making the information society available for everyone in Belgium, and at enhancing Belgium's competitive positions in the information and communication technologies.

The "Five-Star ICT Masterplan" is based on the best practices from all over the world and is built up around these five key success criteria:

- The introduction of "e-government" in Belgium.
- Universal access and internet competencies.
- ICT infrastructure.
- Knowledge and innovation.
- Regulation and law.

We look forward to making a primary contribution in the construction of a future-oriented Europe.

QWE2000 Keynote Session K3-2

Mr. Thomas Drake
(ICCI)

The Future of Software Quality - Our Brave New World - Are We Ready?

Key Points

- Quality for network centric systems
- Software quality engineering
- Future of quality and testing

Presentation Abstract

How will the software quality market evolve over the next few years?

What is at stake? What is it going to take?

Internet, the Web, and e-Business are increasingly demanding higher and higher levels of quality for network-based software systems with less and less mean time between failures. Why?

The impact of poor quality and less than robust systems increasingly affect the bottomline for many businesses.

A lot has happened to improve the situation and new methodologies and new tools for improving software quality have emerged in the last few years, but I would suggest that radical new approaches and initiatives must be created and adopted if the desirable quality levels to support the e-Economy are to be truly realized and especially at the Internet and global level.

What would that look like?

It will take a combination of component-based development, design by contract, specification-based testing, and statistical process control. It will require and even demand much higher levels of predictive and profiling analysis.

It will also demand enterprise level and even systems thinking as more and more complex "web-enabled" applications are deployed into and amongst various market spaces. So we have the twin challenges of faster and faster delivery and deployment times requiring higher and higher levels of quality.

All of these changes are placing great pressure on the traditional ways of testing and viewing quality.

The 'target" customer is no longer just the QA or test group. Increasingly what is demanded is a business and enterprise level focus in addition to the technical.

Moreover, these challenges are not only technical but also cultural in nature and it may be useful to describe at a survey level this "new" future in terms of the new initiatives that are now increasingly required including component-based development and applied software engineering, specification-based testing, test coverage and analysis technology, design for test principles, various predictive software and profiling analysis techniques and approaches, and full life-cycle application testing activities and methodologies as well as a concentrated focus and emphasis on the various quality and testing ôstatesö as part of these initiatives.

The future of quality demands nothing less.

About the Speaker

Mr. Drake is a software systems quality specialist and management and information technology consultant for Integrated Computer Concepts, Inc. (ICCI) in the United States. He currently leads and manages a U.S. government agency-level Software Engineering CenterÆs quality engineering initiative.

In addition, he consults to the information technology industry on technical management and software engineering and code development issues.

As part of an industry and government outreach/partnership program, he holds frequent seminars and tutorials covering code analysis, software metrics, OO analysis for C++ and Java, coding practice, testing, best current practices in software development, the business case for software engineering, software quality engineering practices and principles, quality and test architecture development and deployment, project management, organizational dynamics and change management, and the people side of information technology.

He is the principal author of a chapter on ôMetrics Used for Object-Oriented Software Qualityö for a CRC Press Object Technology Handbook published in December of 1998. In addition, Mr. Drake is the author of a theme article entitled: ôMeasuring Software Quality: A Case Studyö published in the November 1996 issue of IEEE Computer. He also had the lead, front page article published in late 1999 for Software Tech News by the US Department of Defense Data & Analysis Center for Software (DACS) entitled: ôTesting Software Based Systems: The Final Frontier.ö

Mr. Drake is listed with the International WhoÆs Who for Information Technology for 1999, is a member of IEEE and an affiliate member of the IEEE Computer Society. He is also a Certified Software Test Engineer (CSTE) from the Quality Assurance Institute (QAI).

THE FUTURE OF SOFTWARE QUALITY -- OUR BRAVE NEW WORLD -- ARE WE (YOU!) READY?

Thomas A. Drake

Quality Architect/Software Anthropologist

Enterprise Management and Information Technology Consulting

Certified Software Test Engineer (CSTE)

4th International Software & Internet Quality Week Europe 2000

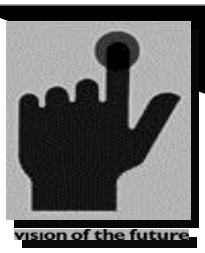
Keynote K3-2

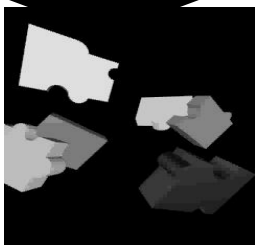
Integrated Computer Concepts, Inc. (ICCI)

http://www.integratedcc.com

thomas.drake@integratedcc.com

Facing the Real Quality Challenge...





Presented at the Speed of Internet

3 hours of info in 45 minutes!

An Historical Perspective...

"The conveniences and comforts of humanity in general will be linked up by one mechanism, which will produce comforts and conveniences beyond human imagination. But the smallest mistake will bring the whole mechanism to a certain collapse."

-Pir-o-Murshid Inayat Khan, 1922
(Tasawuuf leader)

Another Historical Perspective...!

"The accelerating rate of change in society is providing increased pressure on management to incorporate the computer into the planning process. The implementation and use of these systems are, however, extremely difficult from the managerial, not the technical, viewpoint. Computer assisted planning alters the power structure of the organization, changes patterns of communications, revolutionizes decision making, and makes new demands on the database. Moreover, it restructures management roles and general disturbs human relationships by threatening security and imposing new demands for cooperation. Traditional management theory and practices are no longer appropriate, and fragmented organizational perspectives must be abandoned."

- James B. Boulden, Computer-Assisted Planning Systems, 1975.

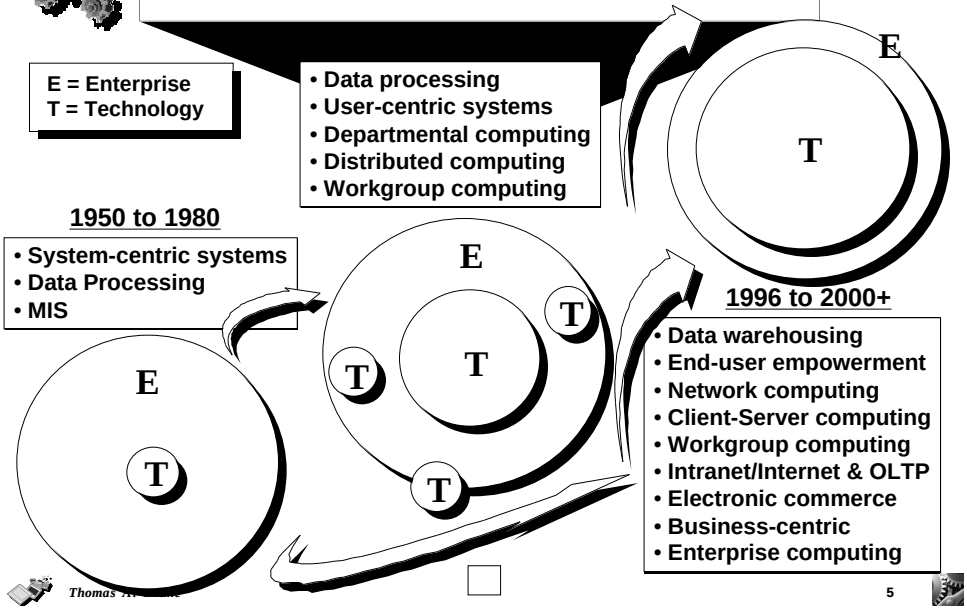
Historical Evolution

E = Enterprise
T = Technology

- Data processing
- User-centric systems
- Departmental computing
- Distributed computing
- Workgroup computing

1950 to 1980

- System-centric systems
- Data Processing
- MIS



1996 to 2000+

- Data warehousing
- End-user empowerment
- Network computing
- Client-Server computing
- Workgroup computing
- Intranet/Internet & OLTP
- Electronic commerce
- Business-centric
- Enterprise computing

Running Rampant in Cyberspace

Missing the Obvious

Forgetting the Fundamentals

Shortened Product Release Schedules

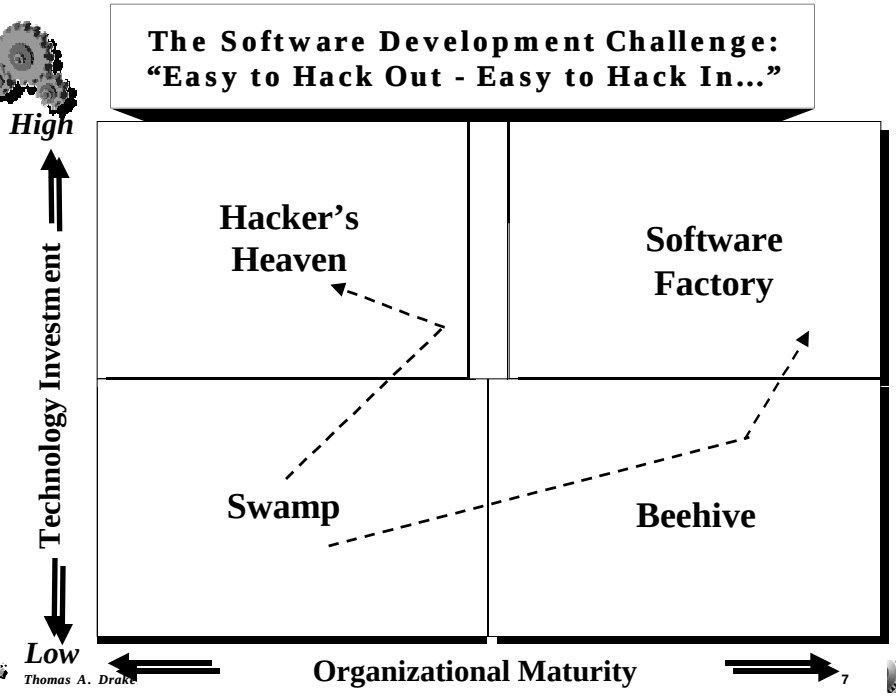
Testing Never Stops when Content is Continuously Changing

Web Time is ALL the Time!

What's Missing "Big Time" ?

Systems Thinking and Perspective!

The Software Development Challenge: "Easy to Hack Out - Easy to Hack In..."



Software - NOT a Naturally Occurring Phenomena!

If you exclude the time it takes to learn, the money that it take to train, the elusive reuse benefits, the resistance to change, the constantly arising trouble spots, the frequent upgrades, the long lead times required to build applications from scratch, the complex programming languages, the lack of scalability, the shortage of talent, the performance penalties, the deployment challenges, the heavy maintenance burdens, the difficulty in comprehension, and the expense of manually reapplying one's customization, then software technology is quite beneficial. :-)

Completely dependent on who we are as human beings! After all it is us humans doing the creating! A product of our fertile imaginations and intellect!

The Challenge for the Enterprise

- ☑ **Boston Consulting Group, Inc. Study - Winter 2000**
- ☑ **100 executives in industries from manufacturing, telecommunications, to financial services**
  - ✓ **Only one of three enterprise management software implementations was successful in terms of cutting costs, meeting business goals and showing a tangible financial impact**
  - ✓ **60% of respondents said the new systems had helped**
  - ✓ **Just over 1/2 said it met their business goals**
  - ✓ **1/3 said software vendors encouraged "excessive" spending**
  - ✓ **12% of the vendors were "fired"**

Sobering Numbers (1)

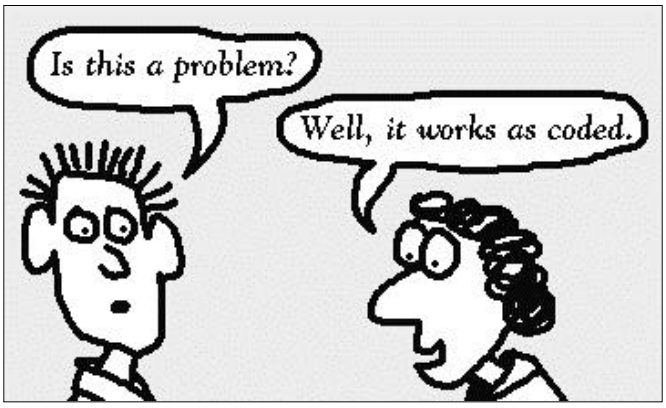
- ☑ **Standish Group Study in late 1995**
  - ✓ **USA spent $81 billion for cancelled software projects**
  - ✓ **$59 billion for projects completed late, over budget, or lacking key or essential functionality**
  - ✓ **Only 16.2% of projects were completed on time & within budget with only 9% in larger companies**
  - ✓ **In larger companies completed projects had an average of 42% of the desired functionality**
- ☑ **Causal Analysis?**
  - ✓ **Lack of user/customer input (requirements not understood/captured)**
  - ✓ **Incomplete requirements and specifications**
  - ✓ **Changing requirements and specifications/requirements creep (+ & -)**

Sobering Numbers (2)

- ☑ **Standish Group study updated 4 years later…**
  - ✓ **Over $250 billion spent annually on IT application development**
  - ✓ **31% of all projects are cancelled before completion**
  - ✓ **88% of all projects are over schedule, over budget, or both**
  - ✓ **For every 100 projects started, there are 94 restarts**
  - ✓ **Average cost overrun is 189% of original estimates**
  - ✓ **Average time overrun is 222% of original estimates**
- ☑ **Even with strong technical skills many project managers and project team members find themselves in over their heads**
  - ✓ **On projects out of control**
  - ✓ **Without the necessary business, organizational and political skills**
- ☑ ***Faith is not a management method*! *Oh, DUHHH*!!**

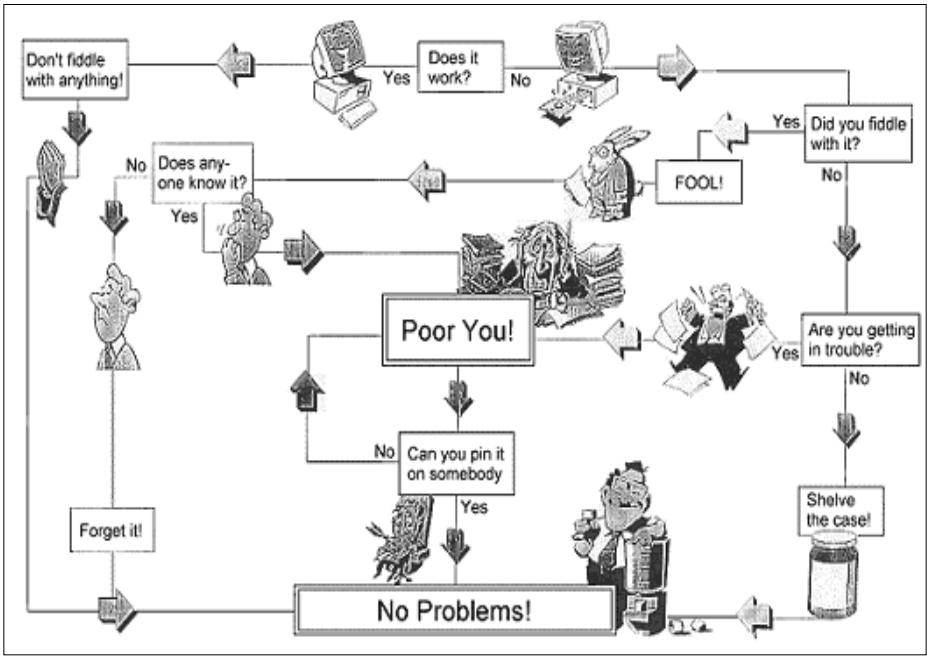
Lack Of Quality - The Epidemic Of Buggy Software

- ☑ **Recently published - *The Software Conspiracy* by Mark Minasi**
- ☑ **We are all guilty! Would you unplug your automated toaster after every 6 slices of bread just to reset the internal software?!**
  - ✓ **The myth that bug-free code is not possible**
- ☑ **Software publishers/contractors STILL aren't generally interested in producing stable, functionally fit, error free software**
  - ✓ **It is features, not quality, that sells! And we buy!**
- ☑ **By hiring the "best and the brightest" we may actually be sabotaging our own efforts - it's embedded in the culture**
  - ✓ **It is the boring but absolutely necessary work that does it!**
- ☑ **Time to emphasize quality - Remember the car industry in the USA in the late 50s, 60s, and early 70s??**
  - ✓ **Also see Jeremy Main's book - *Quality Wars***
- ☑ **And it's the business side of software quality that gets us in trouble!**

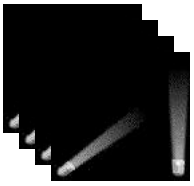


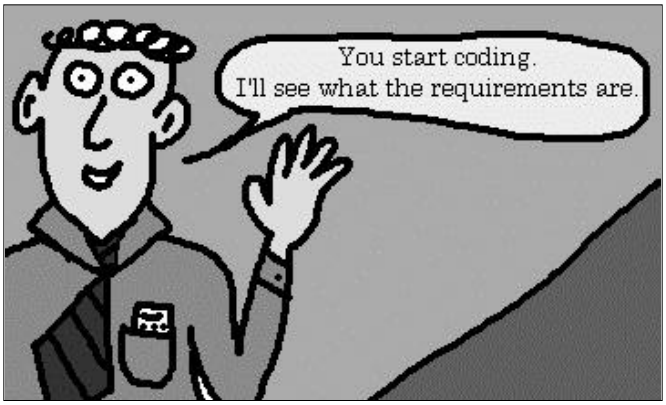
Conventional Wisdom For Software Quality Engineering :-)

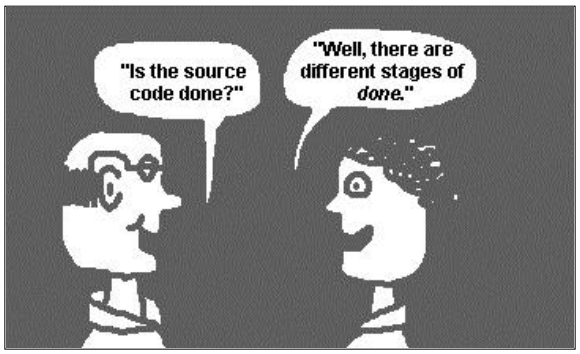
- ☑ **Software Engineering Guidebook on Terminology, v2.0n**
- ☑ **NEW: Different colors from previous version.**
- ☑ **ALL NEW: Software is not compatible with previous version.**
- ☑ **UNMATCHED: Almost as good as the competition.**
- ☑ **ADVANCED DESIGN: Upper management doesn't understand it.**
- ☑ **NO MAINTENANCE: Impossible to fix.**
- ☑ **BREAKTHROUGH: It finally booted on the first try.**
- ☑ **DESIGN SIMPLICITY: Developed on a shoe-string budget.**
- ☑ **UPGRADED: Did not work the first time.**
- ☑ **UPGRADED AND IMPROVED: Did not work the second time.**



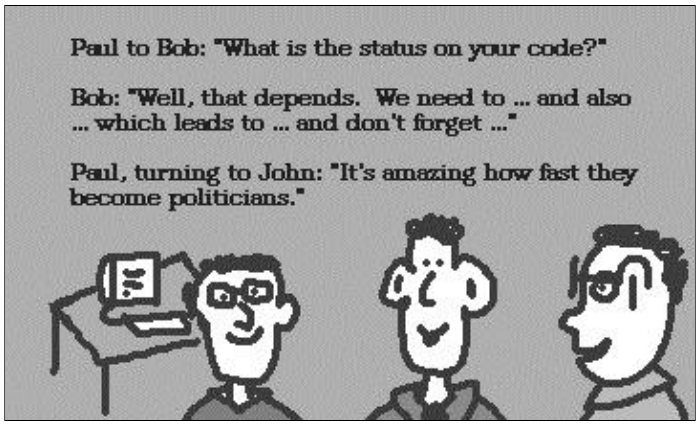
Living in a state of constant ambiguity...







Depends on what you mean by "done" - Or depends on what "is" really is!



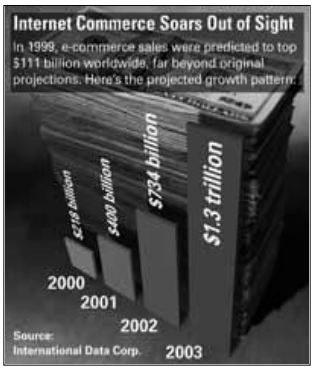
What's Happening?

- ☑ **Dynamic and New Information Sources**
  - ✓ **Global Information Network**
  - ✓ **Diversity of Telecommunication Alliances and Global Arrangements/Alignments**
  - ✓ **New Telecommunication Technologies**
  - ✓ **Explosion of Wireless, IP Telephony, Virtual Numbers and more!**
  - ✓ **Internet and Beyond (Dynamic Roaming)**
- ☑ **New Information Technologies**
- ☑ **New Protocol Structures**
- ☑ **Massive Interdependencies**

- ☑ **Software can be technically correct, but still not succeed**
- ☑ ***All place HUGE demands and MASSIVE Strain on Quality!***

Where Are We Going?

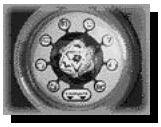
- ☑ **Integrated Component Design and Code**
- ☑ **Software Reuse (domain/pattern level)**
- ☑ **Network Common Services**
- ☑ **XML & Web-Enabled Technologies**
- ☑ **Intelligent Agents**
- ☑ **Data Visualization - (2-D vs 3-D)**
- ☑ **Dynamic Security**
- ☑ **Private/Public Network**
- ☑ **Web-based Workflow**
- ☑ **Extreme Programming**
- ☑ **AutoCode Generation**
- ☑ ***The "Old" QA Paradigms Come Up Way Short!***



The IT Adventure - Software is at the Heart of IT!

- ☑ **Internet/Virtual Reality - Just the Click of a Mouse!**
- ☑ **Surfing the Planet/Real-Time Data Flows**

- ☑ **E-mailing/Electronic Communication and Messaging**
- ☑ **Critical Reliance on Cyberspace**
- ☑ **Satellites/Computer Networks/Digital broadcasting/Cellular/Interactive TV**

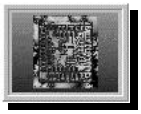


- ☑ **Entertainment and Games!**

Our Reliance on Software is Huge!

- ☑ The network is the program
- ☑ Increased recognition that software is a critical national resource for many countries and vital to the economy
- ☑ Software is THE core technology that supports practically every kind of human activity - Think about it!
- ☑ Can't talk about our lives now, or in the future, without mentioning software. It is ubiquitous!

And Only Getting MORE Connected



- ☑ The future is not what it used to be!
- ☑ Pace of change (Default Standard)
- ☑ Time, communications, space (compressing)
- ☑ Implication - Speed of light access and impact
- ☑ All linked together in one dense, interconnected web of information and data!
- ☑ ***Places HUGE demands on QUALITY***!

Internet/Cyberspace Quality (1)

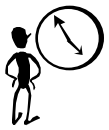
- ☑ **Internet and World Wide Web are the next logical step**
  - ✓ **The Web is huge - 24/7 & 365**
  - ✓ **Part of our culture**
  - ✓ **Look at any ad on television and you will see a Web address**
- ☑ **Endless possibilities for the Web and they go far beyond just static advertising**
  - ✓ **A cybersurfer can find information on almost anything, play games, download software, interact with "live" humans**
  - ✓ **Web is the "Great Equalizer"**
  - ✓ **A one or two person company can look like a multibillion dollar corporation to its customers**

Internet/Cyberspace Quality (2)

- ☑ **Competing for shelf space versus competing for Web space**
  - ✓ **Producing a shift toward the Web and more and more testing of internet-based software**
- ☑ **More complex programs and applications using Java and Shockwave and Flash are emerging**
- ☑ **Testing on multiple platforms and operating systems**
  - ✓ **Different Internet service providers and methods of connecting to the Internet**
  - ✓ **Can't afford to put out a shoddy product on the Web**
- ☑ **Quality is STILL quality and even more so on the Web**
  - ✓ **Stakes are much higher in this kind of "operating" environment and bugs/problems/defects are much more visible**

Quality Makes a Difference!

- ☑ **Issue of market cycle time and shelf life**
- ☑ **Problem of code entropy**



- ☑ **Impact of incremental patching and upgrading**
- ☑ **Doing it right the first time - interface design is everything**
- ☑ **Simple and elegant solutions - stand the test of time and the marketplace**
- ☑ ***Quality is, as quality does!***

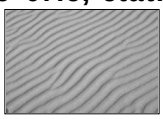
Best Practices - The Use of Domain/Design Patterns (1)

- ☑ **Set of objects with certain known roles and responsibilities**
  - ✓ **Relationship to each other**
  - ✓ **Common usage**
  - ✓ **Prerequisites**
  - ✓ **Cataloged/documented**
  - ✓ **Refinement/updates/extensibility**

- ☑ **Emerging due to Internet time/intense schedules**
- ☑ **Program structures are fundamental**
  - ✓ **Where are the execution cycles**
  - ✓ **"The most efficient instruction set is the one that's never executed!"**
  - ✓ **Provides for the abstraction that provides summary/overview**

Best Practices - The Use of Domain/Design Patterns (2)

- ☑ Patterns should also identify the intended use audience
  - ✓ Provide for the external and internal assumptions
- ☑ Document/Document/Document! (1)
  - ✓ Name, problem, context, constraints, trade-offs, static relationships and dynamic rules/behavior, variants/specializations
  - ✓ Examples - sample implementations
  - ✓ Known uses - describes known occurrences of the pattern and its relationship and application within existing systems



- ☑ Generalize! - How can the program be developed such that it minimizes the code interdependencies among the various subsystems?

(1) Source: Patterns and Software: Essential Concepts and Terminology - Brad Appleton (www.enteract.com/~bradapp/docs/patterns-intro.html)

The Power of Patterns

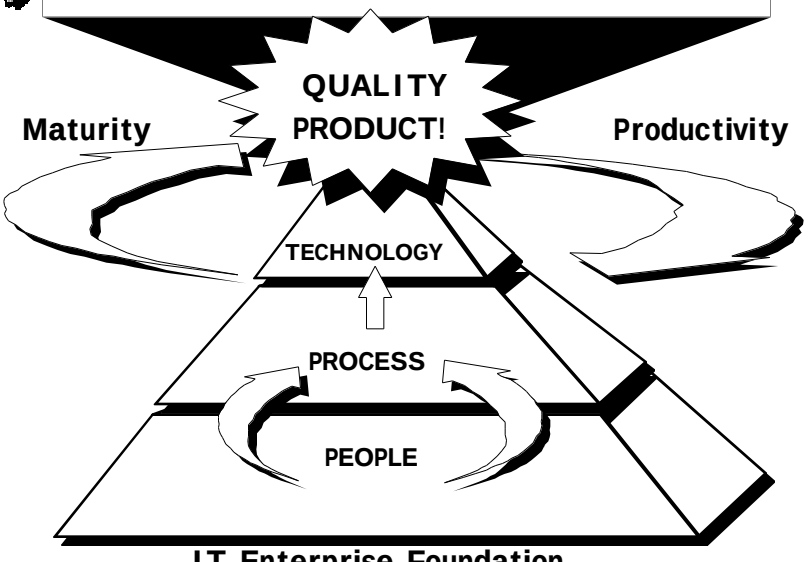
- ☑ Patterns…
  - ✓ Patterns provide a relationship between actions and then "embody" it in a design artifact
  - ✓ Ultimately facilitates the reuse of code and design
  - ✓ If clean design is the most difficult and important step in software engineering, there could be real benefits in adopting

- ☑ Where do patterns come from?
  - ✓ Tangible: Architecture/materials
    Intangible: process/procedures
- ☑ A domain pattern "sensitivity" problem…
  - ✓ Software development culture is still dominated by "bottom-up" thinkers (trees/forests)
  - ✓ For patterns, you need to think "top down" and not just on the bit level but the aggregate of bits (atoms/molecules)

So What Is Software Quality Engineering Really All About??

- ☑ **"...Too many organizations have spent too much time obsessing on the information they want their networks to carry and far too little time on the effective *relationships* those networks should create and support. This is a grave strategic error."**
  - ✓ **Michael Schrage, MIT Media Lab Research Associate, in a white paper of The Merrill Lynch Forum**
- ☑ **The language is not THE answer! But the language is a primary means of communication with its own syntax, structure, rules and meaning.**
  - ✓ **Design by Contract - Precise definitions/relationships**
  - ✓ **And it shows in the Code!**
  - ✓ **Rigor and discipline are fundamental**
  - ✓ **And it is the quality of the software "experience" that may be the real measure of quality!**

Software Quality Engineering - The Heart of Information Technology



Quality Communication It's All About People! (1)

- ☑ Here is the real challenge -
  - ✓ Language of the programmer, software engineer and developer is in terms of data structures and procedures, and network routers and protocols
  - ✓ Language of the end-user, customer, operations support is in terms of general behavior, "state" or use conditions, applications, and operational functionality
- ☑ Two types of information knowledge
  - ✓ Control flows and data structures vs. descriptive
- ☑ Also helps explain the presence of "unstable" business requirements, poor "contract and program management, and overly optimistic evaluations that are largely date driven - all "soft" side issues not technology issues

Quality Communication It's All About People! (2)

- ☑ Most end-users/customers express requirements in terms of natural language (NL)
- ☑ Developer translates NL into symbolic/representational objects making up a domain model
- ☑ And neither the twain shall meet!
- ☑ The "weak link" theory about requirements (the missing link)
  - ✓ Need to trace the human sources of actual requirements
  - ✓ Traceability problem is particularly serious in the later stages of requirements engineering with HUGE impact "downstream"
  - ✓ Impact: Design decision rationale is rooted in the specification, ambiguity here will reflect in the product!

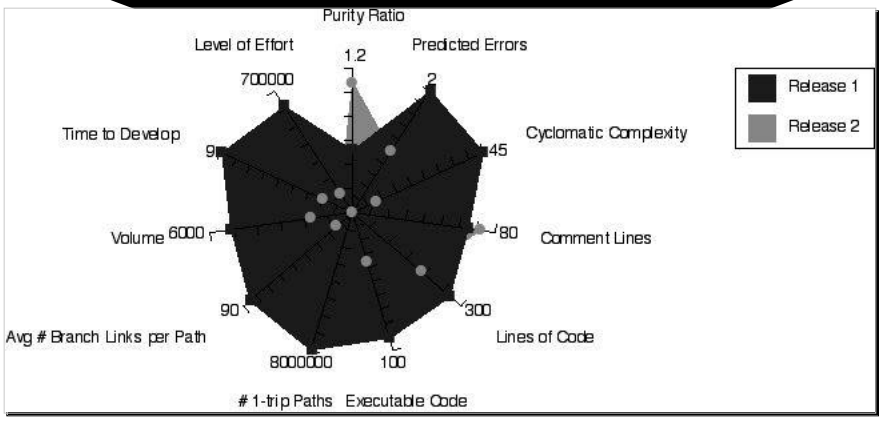
Quality Communication It's All About People! (3)

- ☑ Many design decisions are tradeoffs in the absence of key data and between competing requirements
- ☑ Information behind the design and the conceptual framework or operational understanding is vital
- ☑ Each set of differing stakeholders interprets in light of their own background assumptions
- ☑ Need the opportunity to check that the interpretation is the same as the intention (or different)
- ☑ Organizational and social issues between and among people have the single greatest influence on the effectiveness of requirements communication and engineering process
- ☑ Yes, technology is important, but must dynamically measure

Statistical Process Control Dynamic Measuring/Tracking

- ☑ "Trend" analyze one evolution of software with another
- ☑ Determine "velocity and acceleration" between the key metrics over time - "delta gap analysis" or "variableness"
- ☑ Track addition of new features vs defects via the metrics (correlate) - phase containment effectiveness
- ☑ Use previous release analysis results as "baseline" and regression analyze the subsequent release
- ☑ Understanding how software quality "evolves" and plot trends in improvements over time - map traceability
- ☑ Based on key static and dynamic measures of code and executable attributes
- ☑ Determine maturity and productivity over time - (BOE/ABC)
- ☑ Analyze major and minor releases - continual CM control!
- ☑ Compare similar subsystems, functionality between software products in equivalent domains

Statistical Process & Product Control - It's Competitive Edge!



Modified "spider" Kiviat diagram for graphically showing differences in quality between Release 1 and Release 2

Many other static and dynamic metrics can be mapped and are!

The Future of Quality Into the Looking Glass! (1)

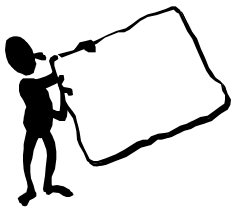
- ☑ **Moving away from large development programs**
  - ✓ **Fixed set of ideas, very difficult to maintain or modify**
- ☑ **Virtual computing -- 24/7, 365 & around the world!**
- ☑ **Component-based development**
- ☑ **Design logic to code!**
  - ✓ **An increasing trend (code engines)**
  - ✓ **Solves many of the "hand" created complexity problems**
- ☑ **Contract-based/specification-based outcomes**
- ☑ **Test and Quality By Design Principles (Design by Contract)**

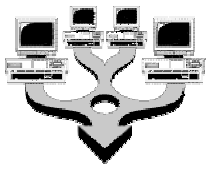
Into the Looking Glass! (2)

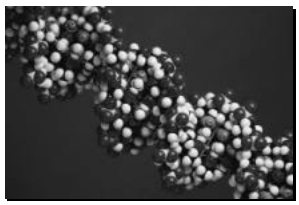
- ☑ **Do not be overly transfixed by C++ or OO or Java or Jini or even the very latest and greatest! Still way too many buzz words and ambiguity!**
- ☑ **Real-time operations across multi-distributed and heterogeneous environments**
- ☑ **The network IS increasingly becoming THE program - just get it!**
- ☑ **Key to software development innovation is business savvy, smart developers, and high quality products that meet and exceed customer expectations and still meet the bottom line!**
- ☑ **Managing in this environment requires a "paradigm shift"**

The "Genetic" Heart and Core of Quality

- ☑ **Design is fundamental!**
- ☑ **Business Rule Algorithms**
- ☑ **Behavioral modeling**
- ☑ **Attributes - The key for quality**



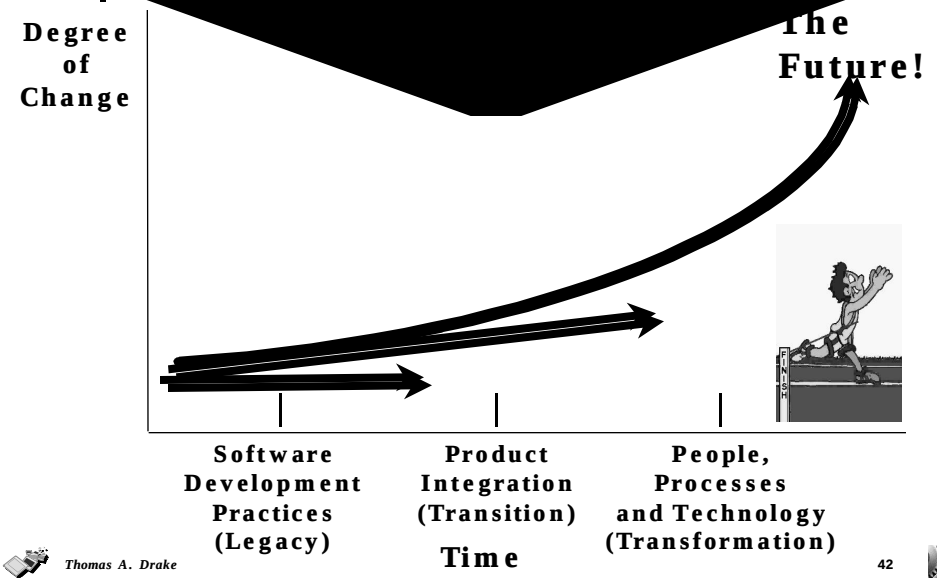




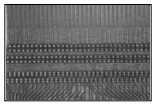
The Continuing Software Quality Challenge

- ☑ Taming the "uncontrolled" distribution of data
- ☑ In software, data is an abstraction
- ☑ Software's strength IS abstraction!
- ☑ Concrete representation vice the abstract notion
  - ✓ How do you build software??
  - ✓ Concept by concept or brick by brick??
- ☑ Agree on the concept!
- ☑ Information hiding is critical
  - ✓ Each module must only access the information it needs
  - ✓ And every software element must have a specification
  - ✓ Use Design by Contract - architecture is critical!

Quality Paradigm Shift Where Do You Live on the Continuum?!



Some Final Thoughts About Quality and Software



☑ **Dealing with multiple levels of complexity**
- ✓ **Quality enemy #1 - orders of magnitude greater**

☑ **Nth Order Computing (modeled after cell life)**
- ✓ **Computer is 1st order - computational!**
- ✓ **Autonomous objects/components are 2nd order**
- ✓ **Interaction of objects is 3rd order - no object alone**
- ✓ **Result/outcome/state transition is 4th order**
- ✓ **Nth Order is shifting function onto structure and code onto data - *Built in dynamic living networks - It's alive!***

☑ ***Computers as machines containing cyberspace containing machines*** **- (Bandwidth, Turnaround Time, and Complexity)**

☑ **So… Are you now ready for this brave new world?! :-)**

Ensuring Data Quality for E-Commerce Systems

The Hidden Role of Data Quality in E-Commerce Success

Eric Messin
Director of Sales
La Grande Arche Paroi Nord
92444 Paris La Defense
France
011-33-140-90-3518
www.vality.com

Quality Week Europe 2000
November 22-24, 2000

E-Commerce: Fast Track To The Promised Land?

- Electronic highway's benefits
  - Instant, global, two-way communication
    - More buyers
    - Seamless ties to partners, suppliers, customers
- Potholes in the electronic highway
  - Easier, "unattended" access makes it more difficult to provide customer service
  - Increased volume of data = increased data variations, misinterpretations and misrepresentations

Quality Week Europe 2000
November 22-24, 2000

Obvious Challenges

- Shoppers' product searches garner
  - Too many irrelevant results; or
  - No results at all
- Customers' free-form data entries not easily linked to back-end standards
- Internal systems' data "not ready for prime time"

E-Commerce: Fast Track To The Promised Land?

- The answer is a qualified "Yes"
- The key enabler is data quality
- E-commerce requires a common, data quality solution – e-quality content management

Defining Data Quality In The E-Commerce Context

- Data quality misconceptions
  - Data is either right or wrong
  - Data is either correctly or incorrectly represented
- Misconceptions oversimplify the fact that perception of data quality varies by user and usage

Data Quality Demystified: An Example

Name	*Street*	*City*	*Zip*
DEC	216B Old County Rd	Lincoln	01773-4603
Digital Equipment Corp	216 Old County Rd Ste B	Lincoln	01773-4603
Compaq	216 Old County Rd	Lincoln	01773-4603

- Order fulfillment, shipping, direct mail ✓
- Marketing ✘
- Contract administration ✘

Data Quality Demystified

- Data quality not an absolute state
  - It is a relative assessment of the degree of usefulness and reliability for an intended purpose
- The more user communities and business processes served by the same data, the greater the risk

Data Quality Is Multidimensional

- Data quality has a time dimension
  - Data values that are correct when recorded may become liabilities in the future
    - Marriages
    - Mergers
    - Zip code reassignments
    - Area code expansions
  - Historical representations often have intrinsic and legal "correctness" that must be preserved

Data Quality Is Multidimensional

- Data quality is contextual
  - Data in internal systems may be inappropriate to expose to a wide public
    - Special discounts and negotiated terms
    - Confidentiality
    - Internal comments
      - Data collected to assist delivery personnel may alienate customers if exposed – "Mean dog in back yard", "stone deaf: holler at them"

Data Denial

- Data quality problems have always existed
- Problems have been
  - Masked or managed by business logic of legacy applications
  - Hidden by an intermediary
- E-commerce "pierces the veil" of operational data

E-Commerce Trends Accentuating Data Quality Issues

- Breadth of user exposure
- Diversity and volume of source data
- Disintermediation
- Privacy and ethical issues

Breadth Of User Exposure

- E-commerce is extending the exposure of the relative nature of data quality
- Older operational data may be unclear or inappropriate, particularly to user whose perspective may be different than anticipated

Example: 14-gauge GRN cable

Is it 14-gauge green cable or grounded cable?

Diversity And Volume Of Source Data

- Transactions must be quick and convenient
  - Minimum of edit checking and verification
  - Web data not easily reconciled with operational data for analyses
    - May even deliberately circumvent fraud and abuse of purchasing policies

Example: Steve Brown, S Brown, Stephen Brown, Steve Bron – Same customer? On credit hold?

Disintermediation

- E-commerce has a self-serve business model
  - Positives: Speeds transactions, saves money
  - Negatives: Loss of buffer or translator, discrepancies left unresolved, loss of revenue

Example: I search for pants, you sell slacks. An intermediary would automatically make the translation and the sale – not any more!

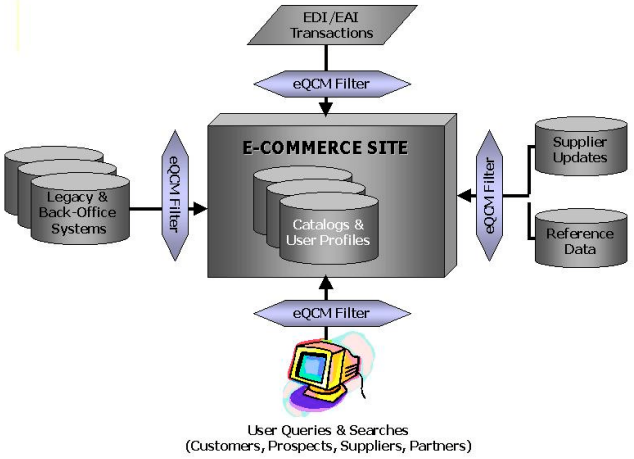
Privacy And Ethical Issues

- Misuse of private data is gaining increasing publicity
- Not just a legislative issue – it's also a data quality issue
  - Incorrect social security # reveals information
  - Inability to identify unique customer reveals financial data

The E-Quality Content Management ("eQCM") Process

- When does it occur?
  - Anytime that data is in motion
  - Not a one-time "clean-up"
  - Real-time data quality maintenance filters are required
  - eQCM must mediate the daily activity that goes against your enterprise databases

eQuality Content Management Control Points



What Processes Are Involved?

- Context mediation
  - Determination of business meaning of word or value based on context
- Fuzzy searching
  - Ability to perform quick and product retrieval of data without a precise key
- Fuzzy matching
  - Application of your business rules, and measure of statistical certainty, to determine when to declare critical relationships within the data, such as duplication and affiliation

What Does It Cost?

- Costs vary depending on the complexity of the data and affected business processes
  - Costs escalate in proportion to amount of risk/reward
    - High-dollar transactions necessitate very high data quality
- *WARNING*: One size does not fit all
  - For example, shrink-wrapped "data-cleansers" designed to prepare bulk mailings will not provide data quality for sales, marketing and CRM efforts

Conclusion

- You must have a requirement to manage data quality whenever:
  - Data is re-purposed or deployed for new business purposes
  - External data sources are integrated, linked or queried
  - Transactions of external origin are synchronized with your Web-site or back office
  - Users (prospects, customers, partners) search your published databases

Ensuring Data Quality for E-Commerce Systems

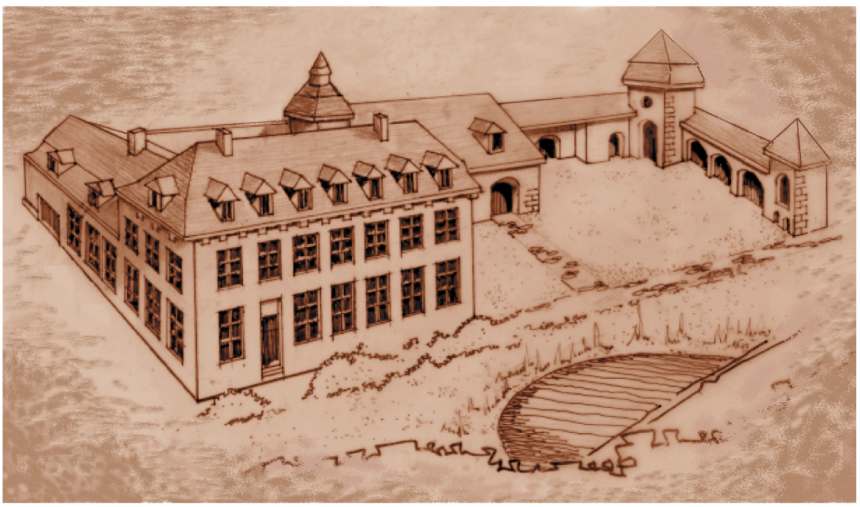
The Hidden Role of Data Quality in E-Commerce Success

Eric Messin
Director of Sales
La Grande Arche Paroi Nord
92444 Paris La Defense
France
011-33-140-90-3518
www.vality.com

Quality Week Europe 2000
November 22-24, 2000

Have a taste of Quality

ps_testware offers a total concept of services under the form of consultancy, coaching and training in order to solve problems regarding software testing.



This is Testing Hill™ Campus, the headquarters of ps_testware. The castle and the surrounding domain reflect the quality we strive at in all our projects.

Testing Hill™ Campus, however, is more than our company address. We want to create a forum for all software testing professionals, co-workers as well as customers and partners. The conference and training rooms are at the disposal of everyone who wants to talk about software quality and structured testing. But primarily Testing Hill™ Campus is the location of the Test Laboratory and the ps_testware Institute (training and R&D centre).

The knowledge gathered by the ps_testware Institute is regularly updated and nourished by our consultants. Once a month they gather at the campus to work on a project, challenge or problem. These fruitful Mondays are called 'Home Sweet Homes'. It is rare that a company invests so much time in its co-workers to brainstorm or elaborate on ideas but we realise that our co-workers are worth the effort.

ps_testware
Software Testing Services

Tiensesteenweg 329 - 3010 Leuven (Belgium)
Tel.: +32 (16) 35 93 80 - Fax +32 (16) 35 93 88
e-mail: info@pstestware.com - http://www.pstestware.com

You'll encounter few objections when communicating the advantages of TesCom. Such is our standing as the world's leading e-testing company and Europe's largest independent software testing solutions provider.

In a huge global market with an insatiable appetite for our top-quality services, we bring new generation software development projects to fruition across Banking, Financial Services, Telecomms, Retail and Utilities - for clients that include Microsoft, JP Morgan, Tesco, the BBC and Cable & Wireless.

Tried, tested and totally focused on even greater success, this is your opportunity to lead our expansion and market a world-class capability built on leading-edge platforms, technologies and software. If you have the technical empathy and track record in a successful team in a consulting sales environment, find out more about our entrepreneurial culture that rewards results and initiative.

Contact Helen Brudenell on 020 7250 4705, fax:020 7250 0464, email: helen.brudenell@tescom-intl.com

TESCOM

the art of testing

www.tescom-intl.com

TesCom European Head Office
5 St. John's Lane, London EC1M 4BH.

SOFTWARE QUALITY ASSURANCE

Amphora Quality Technologies

Amphora Quality Technologies (AQT) is a high-tech company that provides the most comprehensive and effective Software Quality Assurance (QA) and Testing solutions to its clients. AQT supplies services that significantly reduce the client's risk. AQT helps you to develop high-quality software that is delivered on time and on target, meets current business requirements and is scalable to future needs.

AQT believes that cost and performance are no longer the main problems of the modern IT industry as was the case several years ago – now, the problem is software complexity. Nowadays, the complexity of information systems is a decisive factor in software quality. That is why AQT guarantees Software Quality, supplying the latest high standard Testing and Quality Assurance services to Developers and Corporations worldwide.

Structurally, AQT consists of four divisions:
· Web Lab – a laboratory specializing in quality of Web Site and Internet applications;
· Functionality Lab – a laboratory for software functionality testing, software test results and source code analysis center;
· Performance Lab – a laboratory for the analysis of software performance characteristics and reliability;
· Research department – the company research center;

Services

Amphora Quality Technologies provides complete Software Quality Assurance solutions to its clients, rendering a full range of Consulting, Testing and Analytical services. Other than the traditional Functionality analysis (black box testing), AQT offers a wide range of special QA services, such as:

· Consulting;
· Web site and Internet application testing;
· Performance and Load analysis;
· Reliability testing;
· Security analysis;
· Middleware server/component testing;
· Communication and Integration analysis;
· Usability testing;
· Technical requirements analysis;
· White Box testing & Source Code analysis;
· Cause Analysis and interpretation of test results.

See www.in-amphora.com/aqt for detailed information.

CMG plc is a leading European ICT services group, providing business information solutions through consultancy, systems development, software applications and managed services. Established in 1964, CMG operates internationally from its European bases in the UK, The Netherlands, Germany, France and Belgium, implementing and supporting applications for clients around the world. The Group is listed on the London and Amsterdam stock exchanges.

CMG has proven knowledge, products and solutions in the finance, trade transport & industry, telecommunications, media, energy, utility and government markets. The Group also provides managed information services ranging from payroll processing and personnel administration to call centres and networks.

CMG works alongside its customers to generate success for them. Through the quality of its services and the long term commitment of its staff, CMG adds value to its customers' businesses and thereby creates a basis for the Group's long term success. You can find more information on CMG at <http://www.cmg.com>.

About TestFrame

TestFrame, developed by CMG, provides a well-documented software testing method that has proven itself in practice. Organisations can easily integrate TestFrame into their current ICT environment. Examples of use are the thorough testing of online systems, web applications, APIs, batch systems, client/server solutions and embedded systems. TestFrame enables organisations to control testing routes better, to enhance the effectiveness of the testing and to improve the maintainability of testing products.

elementool, Inc. is the leading Application Service Provider of Web based software bug tracking and support management tools. elementool provides its tools to software companies and business web sites from all over the world. elementool is used on a daily basis by its customers, and is integrated in their product development process. Benefits of the elementool tools: no installation of software or hardware is required - it is only one click away, application is available from every location in the world using the Internet, customizable forms, powerful reports, enables the submission of issues by your customers directly from your web site, email notifications, self-download of issue list for off-site backup and much more. Logon today to our web site for demo: http://elementool.com

eValid ™ -- The Internet Quality Authority ™
Client-Side Browser-Based WebSite Quality Checking,
Testing, Validation, Tuning, Loading, 24x7 Monitoring
Training, Consulting, Seminars

About eValid -- The Internet Quality Authority

eValid enhances your e-business success by assuring that your WebSite is trouble-free, reliable, speedy, and available 24x7. In a Web-paced world your WebSite is your key asset. eValid checks, protects and insures.

eValid Products

eValid's Test Enabled Web Browser™ is a test engine that provides you with browser based 100% client side quality checking, dynamic testing, content validation, page performance tuning, and webserver loading and capacity analysis.

This new cutting-edge technology, is 100% client side based, and is completely object-oriented. eValid offers a unified approach to WebSite testing that is unique in its simplicity, power, efficiency, effectiveness, and superior ease of use.

By focusing entirely on the users' view of WebSite quality, eValid results are accurate, complete, repeatable, and highly effective -- all as experienced by your users. The eValid test engine is available in several product configurations.

eValid Services

eValid website quality services are all based on the eValid test engine, and are are supported through training, consulting, and technical seminars.

- **Standard Monitoring**: eValid monitoring, based on the eValid test engine, runs standard tests on your site. eValid's 24x7 website performance monitoring provides for email and/or pager/beeper alert service, plus customer access on our WebSite to historic testing and monitoring data. *Be the first to know whenever your site is misbehaving.* More...
- **Custom Monitoring**: Use eValid test services to contract us to run tests you have recorded and proved out yourself using the standard eValid test engine. Custom eValid test executions run on standard intervals, in varying time zones, and are all 24x7. *Make sure your **own** tests run successfully all the time.* More...
- **WebSite Testing, Qualification, Verification, Loading**: eValid consulting services include WebSite testing, test suite development,

- **Testing:** eValid test scripts can exercise the key parts of your site, confirm links, check content, and simulate users' activities. *Make sure your customers get the right message!* More...
- **Validation:** eValid can confirm selected content, validate document properties, images and applets. *Have confidence that you are delivering correct information!* More...
- **Tuning:** eValid timing capabilities let you identify slow-loading pages so you can "tune up" your site for optimum performance. *Keep customers from clicking away!* More...
- **Loading:** eValid load testing scenarios can simulate 100's or 1000's of users. *Can your WebSite handle the traffic when a serious crunch comes?* More...

WebSite qualification, e-commerce verification, and WebSite loading and capacity checking exercises. All work is based on application of the eValid test engine plus other non-released WebSite analysis facilities. More...

- **WebSite Quality Consulting & Seminars:** eValid website quality experts can work along side your web developers to make sure your site meets the highest reliability, quality, performance, and capacity standards. eValid seminars and workshops are aimed at bring your own team up to speed. More...

eValid -- Your E-Business Partner

eValid -- offering products and custom services -- is your one stop solution provider for WebSite quality. eValid is your true e-business partner.

eValid, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA

Phone [+1] 415.550.3020
FAX [+1] 415.550.3030
info@soft.com.

gitek®

interaction through software

Gitek nv

Gitek nv was founded in 1986 and is in addition to the design and development of software, specialised in software testing. Gitek employs more than 120 people, with 40 professional software Test Engineers. Its customers include the pharmaceutical industry, the financial industry, the telecom industry and insurance companies.



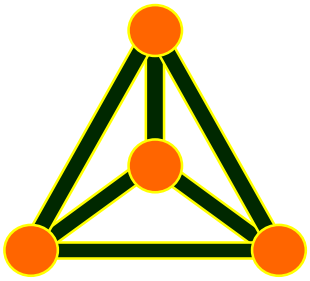
The difference

Gitek is an enthusiastic team of professionals, specialised in customised IT solutions and structured testing. A personal approach forms the basis of our success. This ensures better-adapted services to the customer's specific requirements, and a committed and motivated team ensuring optimal efficiency.

Structured Testing

Gitek is exclusive distributor of TMap® (Test Management Approach), TPI® (Test process Improvement) and TAKT© (Knowledge of Testing, Automation and Tools) in Belgium.
Gitek provides services that offer a complete solution for testing:

- Participation in the operational test process
  - Test planning and test management
  - Test design and test execution
- Complete test projects and fixed price projects
- Test advice and support
- Defining and implementing a structured test process
- Selection and implementation of test tools
- Improvement of the test process
- Training and coaching in testing



Contact us

Gitek nv
Sint Pietersvliet 3
B-2000 Antwerp (Belgium)
http://www.gitek.be
e-mail: gitek@gitek.be

IDEA TO BUSINESS

INNOVATION MANAGEMENT SERVICES

CONSULTANCY FOR THOSE WHO CHANGE THE WORLD

I2B *(Idea to Business) is a new Belgian consultancy company founded in 2000 by six people of which five are experienced consultants. Their consolidated know-how and skills have resulted in a complete portfolio of competences required to run projects concerning ICT, E-Commerce or Innovation (new business development). Together they result in* ***Innovation Management Services,*** *offered to two types of clients:* ***Large Enterprises*** *and* ***Small & Medium Sized Enterprises (SME's)****. For the latter, I2B developed a special delivery model allowing SME's to receive expert knowledge to which normally only big organisations have access to.*

The Mission of I2B is:

To assure that companies can innovate and realise sustainable business from their ideas.

INNOVATION MANAGEMENT SERVICES

We live in an era of Innovation. Product life cycles are shrinking; new technologies are developed faster. It becomes a real challenge to meet deadlines and to make it to the market before competition does. To survive in this extremely dynamic world, companies face the paradox of having to master a variety of competences (gaining experience) whilst continuously changing and improving (learning new technology).

I2B provides its competences, listed aside, through a team of experienced consultants. We help you improve on your **Key Performance Areas** using a combination of these skills.

We basically work in two main area's. **ICT Engineering** and **Strategic Innovation**. In the first area we provide a range of tactical services that will help you in realising successful projects in ICT or **E-Commerce**. Amongst the many issues of ICT projects, assuring quality is the most difficult one. The backside of this flyer gives some examples of situations you might be in.

The second area concerns **innovation**. A lot of companies struggle with their legacy. They have difficulty in catching up with the new technology and they find themselves losing market share from small new and dynamic entrepreneurs. These companies need agility if they wish to survive in the future.

We provide training, coaching, operational and management consultancy

Our competences:

- Structured Testing
- Quality Assurance
- Requirements Mgt.
- Release Management
- Joint Engineering
- Project Management
- Change Management
- Knowledge Mgt.
- Innovation Mgt.
- Human Resources

Company/Product description
for QWE '2000 Expo Guide and CD-ROM Proceedings

McCabe & Associates (UK)
Chancery Court
Lincoln Road
High Wycombe
Bucks
HP12 3RE
Tel: +44 (0) 1494 463 233
Fax: +44 (0) 1494 463 288
Email: sales@mccabe.co.uk
http:// www.mccabe.com

McCabe & Associates is an international leader in software solutions for improving the quality and reliability of enterprise software applications. Based on over twenty years of research and experience in software quality, testing and re-engineering, **McCabe IQ**™ is an integrated approach to building quality into your software development lifecycle. McCabe IQ combines a strong theoretical foundation with practical, visual tools to help organisations:

- Accurately assess software quality, complexity, and testing requirements
- Pinpoint potential problems and high-risk areas
- Eliminate redundant code
- Thoroughly test applications
- Validate coverage of high-risk areas

McCabe IQ™ supports developments in Java, C++, C, VB, COBOL, FORTRAN, PL1 and Ada

Since 1977, we have been working closely with our customers to improve the quality of large-scale, mission-critical software. Many of the worlds most influential corporations and government organisations have used our products successfully to test, re-engineer, and verify the quality of over 30 billion lines of mission-critical code. Today, we are one of the most highly respected vendors of products for source code analysis and testing. Our structured testing methodology has been adopted and published by the National Institute of Standards and Technology (NIST).

Solidly based on source code analysis technology, McCabe IQ integrates the build, test and change phases of software development. By linking these processes through advanced visualisation, industry standard metrics and dynamic monitoring, McCabe IQ raises the quality standards of your deliverables, reduces your overall development costs, decreases your time to market and lets you focus your resources where they will have the greatest impact.

Adrian Cowderoy

ProfessionalSpirit

+44(UK) 118 9 427 970

46 Western Alms Avenue

Reading RG30 2An

United Kingdom

ps_testware
Software Testing Services

Tiensesteenweg 329 • 3010 Leuven • Belgium
Tel.:+32-16-359380 • Fax:+32-16-359388
E-mail: info@pstestware.com • http://www.pstestware.com

---

ps_testware is a privately held company, active in the world of Information Technology. We provide Software Testing Services to the IT market.

In a business concerning software quality, ps_testware has chosen a qualitative approach over a quantitative. We provide our services through a team of highly skilled Management Consultants, Test Consultants and Test Engineers. Using our years of experience, we help you to improve your software processes and assist you with our knowledge of Structured Software Testing. Not only can we advise you on the right procedures, we can also help with, or perform, the tests you need to assure software quality to your customers.

Through our unique services, we can not only assess your testing department but also your complete IT department. If necessary, advice can be given to structure your processes and procedures.

Next to this, we train our customers in the methodology of Structured Software Testing. Instead of just offering you a stand-alone training, we have created three profiles. Depending on the function and responsibilities of a person, we provide the correct training. This way we can ensure a shorter learning curve and less coaching on the job.

Through our outsourcing services, we test for our customers. We can even create a complete test laboratory on site or at our own offices.

If you need expert knowledge in Software Testing Services, don't hesitate and contact us or visit our website: http://www.pstestware.com.

PolySpace Technologies Invites You at Quality Week

A Quantum Leap in Software Verification

PolySpace Technologies provides today a groundbreaking solution to drastically improve software reliability of Ada and C applications.

The PolySpace Verifier product aims at statically and automatically detect run-time errors in Ada and C critical software applications. It allows substancial verification costs reduction and increase quality to a level which was simply impossible to attain before.

The PolySpace Verifier product was first experienced after the Ariane 5 European Launcher crash and is now used in avionics, space, energy, automotive and railways transportation sectors.

Technical addendum:

PolySpace Verifier relies on a breakthrough in Static Verification Technology based on abstract interpretation. It's a radical departure from existing and debugging tools. Instead of dynamically and iteratively verifying software consistency, it statically, exhaustively and automatically discovers sections of code that are sources of run-time errors including attempt to read non-initialized data, null and out-of-bounds pointers dereferencing, out-of-bound array accesses, float and integer overflow, illegal type conversion, arithmetical exception, division by zero and concurrent accesses to unprotected shared data.

If you are involved in critical software development you'll be sure interested by our technology. Complete now your development process by using PolySpace Verifier: You will reduce your validation costs and increase the quality and the reliability of your applications without changing anything in your way of working.

Radview Software Ltd.

Corporations are gaining a competitive edge by investing broadly in Web technologies including E-business systems that link them with customers, employees and partners. Businesses and business models are evolving to fully leverage the potential of Web technologies, and are dependent upon maximum availability and performance in their line-of-business systems to be successful.

RadView Software specializes in developing Web application testing solutions. Our mission is to deliver best-of-breed scalability and integrity testing tools developed specifically to meet the evolving needs of Web developers, testers and IT resources. Our vision is to provide high-performance, easy-to-use and cost-effective testing solutions to a broad market. RadView introduced its award winning, flagship product WebLoad, in November 1997 and it quickly emerged as the premier Web application testing solution.

WebLoad is the only testing tool that unifies load, performance and functional testing into a single process for shortened development cycles and unmatched verification of scalability and integrity prior to deployment. WebLoad verifies Web application scalability by generating a load composed of Virtual Clients that simulate real-world traffic. Users create JavaScript-based test scripts that define the behavior of the Virtual Clients. WebLoad executes these test scripts and monitors the Web application's performance providing real-time graphical and statistical results and comprehensive reports.

WebLoad Resource Manager extends comprehensive testing and analysis capabilities throughout the application lifecycle, improving organization collaboration and enabling verification of application quality. Through a single, standard solution, it allows organizations to pool Virtual Clients and hardware providing faster issue resolution and allowing everyone in the development team to contribute to a successful deployment.

For additional information about the company, contact radview at +972-3-765-0555, e-mail: info@radview.com or visit us on the internet at: http://www.radview.com

Rational
the e-development company

Rational Software helps organizations develop and deploy software for e-business, e-infrastructure, and e-devices through a combination of software engineering best practices, tools, and services. The Rational e-development solution helps organizations overcome "the e-software paradox" by accelerating time-to-market while improving software quality. This unique, integrated solution simplifies the process of acquiring, deploying and supporting a comprehensive software development platform, reducing total cost of ownership.

IDC has recognized Rational as the market revenue leader in multiple segments of the software development life-cycle management market for four years in a row.

Process Made Practical

The implementation of a predictable, repeatable process is crucial to software development success. That's why Rational developed the Rational Unified Process™, a set of software engineering best practices optimized for the rapid development of quality software. The Rational Unified Process integrates with Rational tools to provide seamless, browser-based delivery of best practices. Adopted by IBM, Microsoft, Sun, and thousands of other leading development organizations, the Rational Unified Process is widely recognized as a defacto industry standard.

Unified Tools for the Team

Rational tools enable team members to efficiently perform their roles and share their work throughout the project lifecycle. Individually, Rational tools are leaders in their respective market categories. Combined, they offer unprecedented automation and ease of use. Only Rational offers a unified set of tools that fully supports individual practitioners, and fully supports the team. Rational tools span Windows® NT and UNIX platforms, and support a host of languages, IDEs and operating environments, including: C/C++, Java, EJB, XML, COM/+, and CORBA.

Key products:

Rational offers over 40 products, which can be purchased individually or in integrated Suite editions. The Rational Suite™ product family, introduced in December 1999, provides a convenient and cost-effective way for companies to acquire a complete e-development platform on one CD-ROM. Each Rational Suite Studio combines one or more practitioner tools with a team unifying platform that enhances collaboration across the entire team.

The Rational Suite Product Family:

- *Rational Suite AnalystStudio™* – for system definition
- *Rational Suite DevelopmentStudio* – for software development
- *Rational Suite DevelopmentStudio RealTime* – for real-time and embedded software development
- *Rational Suite TestStudio™* – for functional and reliability testing
- *Rational Suite PerformanceStudio®* – for functional, reliability and performance testing
- *Rational Suite Enterprise* – for the practitioner who spans multiple functional areas

Rational®
the e-development company™

Our mission is to be the industry leader in providing customer/market driven innovative solutions, using software and services, to meet the requirements of product development, on a worldwide basis.

SDRC's e-PKM software is a suite of solutions that helps you store and archive product and process data within your entire enterprise. It enables you to reuse that data, creating information in a digital process. As information is reused -becoming knowledge- your organization can move into higher levels of competitive excellence.

SLATE™ is a systems engineering solution for capturing "the voice of the customer" by way of product requirements. SLATE supports Integrated Requirements Management. This means the requirements are integrated with the entire product development process--allowing requirements to influence the design decisions of all disciplines, which tends to guarantee compliance with customer desires. The result is that more compliant products are built on schedule and under budget in general, rather than as the exception.

SLATE™ is a groupware solution that enables companies to apply a first-of-its kind set of systems engineering capabilities to product development, in a client/server, multi-user, repository based environment. The SLATE product line is divided into different modules, with each one able to access the same SLATE database providing different capabilities:

- SLATE REquire™ is a requirements engineering and management system. REquire enables system engineers to make architectural trade-offs in the context of customer requirements. REquire provides a wide range of capabilities including document management and requirements engineering.
- SLATE Architect is the full-featured SLATE system. It contains all system modeling, requirements engineering and document management functions.

SLATE™ integrations with UML CASE tools provide the ability to trace structured software requirements to UML Software system elements and their test results.

SDRC SLATE™ development strategy is to extend conventional user interfaces, such as MS Office applications and Visio, with SLATE U/I functionality which can be used to let engineers work with the SLATE database using a particular view depending on their role in the organization.

SDRC Nederland BV
Rivium Quadrant 81
2909 LC Capelle a.d. Ijssel
Marsh & McLennan Bldg
The Netherlands
Tel: +31-10-44-72-088
Fax: +31-10-20-23-434
http://www.sdrc.com

The SIM Group

The consultancy for efficient testing services

SIM has been professionally testing software for more than 11 years and has a proven reputation for success in testing and, most importantly, the projects and systems SIM has tested. SIM's approach to testing follows the good practice established and standards encountered during this history of successful testing. All of SIM's testing work involves the best possible use of Automated Testing techniques, an area in which SIM has a 100% success rate.

The SIM approach to testing makes the best possible use of testing technology and the most effective use of professional and well-qualified testers. The result of a SIM testing project is efficiency, effectiveness and excellence. SIM has a team of approximately 70 permanent staff based in the UK.

SIM TestLab – SIM have a TestLab staffed with a team of testing professionals which operates 24 hours a day, 7 days a week. SIM are able to provide a variety of services from our TestLab which include:

- Functionality, regression and penetration testing
- Compatibility testing
- Usability testing
- Load, performance and stress testing
- Continuous monitoring of your site to ensure your site is available 24/7

SIM Services – SIM provide other high quality services which are all aimed at efficient and effective testing. These services range from short term consultancy projects, fully managed testing projects through to outsourcing of testing.

SIM's independence - SIM's independent position on all issues related to testing is invaluable to its clients. SIM provides an objective view of the test processes taking place and can recommend the tools, utilities and technology best suited to each environment. SIM's working relationship with major tool vendors - such as Mercury Interactive, Segue, McCabe and Associates, Rational, Compuware and RSW, and experience in the implementation and use of such tools - places the Group in an unparalleled position for independent advice.

Methodologies – A documented testing methodology is essential for system delivery. SIM develops and documents the procedures, methods and material for formal, structured testing to meet the specific needs of each client. SIM has also developed its own approach to Automated Testing Support called tMosaic - an automation process unique to SIM - based upon an already proven combination of techniques, tools and training. *t*Mosaic is effective on all systems and platforms and has the flexibility to be customised to support existing methodologies already in use at client sites.

Contact:
The SIM Group Ltd
White Rose Court
Oriental Road
Woking
Surrey
GU22 7PJ
Tel: +44 (0) 1483 740289
Fax: +44 (0) 1483 720112
http://www.simgroup.co.uk/
Email: info@simgroup.co.uk

TestWorks

Software Research, Inc.'s TestWorks, an integrated suite of software test tools, is the broadest test tool suite available. TestWorks tools help automate and streamline the software development and testing process with product lines that work independently or as an integrated toolsuite. TestWorks is the only tool suite that offers Regression Testing, Test Suite Management, and Test Coverage support for Web **and** Windows **and** UNIX Platforms.

TestWorks Overview

You can get most of your basic questions about TestWorks products answered from the TestWorks Frequently Asked Questions (FAQs).

Send Email with your question to info@soft.com and we're respond quickly.

TestWorks for Web

TestWorks/Web™ is a bundle of software test tools tailored to support complete regression testing for Web Sites, including those that exploit the advanced features of Java™.

The **TestWorks/Web** product bundle consists of three major components:

- eValid for Windows
  - Frequently Asked Questions(FAQs)
  - Take a Tour of eValid
  - Features and Benfits Summary
  - Pricing and Order Form
- SMARTS (hierarchical test controller)
- TCAT for Java/Windows (coverage analyzer for Java)

Read the new White Papers by Edward Miller on The WebSite Quality Challenge. and WebSite Testing.

Consider one or more of our new subscription-based eValid Test Service Options that analyze your site's security, reliability, content quality and performance.

TestWorks For Windows

TestWorks for Windows, an integrated suite of automated testing tools, is the broadest suite of tools available to test applications running under MS/Windows (Win3.1), MS/Windows NT or MS/Windows '95/'98 (Win32).

Testworks for Windows has two main bundles of tools:

- TestWorks/Regression
  - CAPBAK/MSW
  - SMARTS/MSW
- TestWorks/Coverage
  - TCAT C/C++
  - TCAT for Java/Windows

TestWorks For UNIX

TestWorks for UNIX is designed to work independently or as an integrated tool suite to provide an efficient, automated testing environment for most UNIX-based

platforms.

TestWorks for UNIX consists of three product lines:

- TestWorks/Regression
  - CAPBAK
  - SMARTS
  - EXDIFF
- TestWorks/Coverage
  - TCAT C/C++
  - TCAT for Java/UNIX
  - TCAT/S-TCAT Ada/f77
- TestWorks/Advisor
  - METRIC
  - TDGEN
  - STATIC

Your Quality Process

TestWorks products support a multi-filter ***Quality Process Architecture***™ for software development projects in C, C++, Java, plus Ada and F77.

The ***TestWorks Quality Index*** is a quantitative Quality Index (Figure Of Merit) that characterizes your Application Development Process.

Downloading Products

Download Datasheets

DOWNLOAD PRODUCTS

License Key Request

QuickStart Manuals

User Manuals

Platforms Support

Check **Available Bundles, Products, Platform/OS, and Versions**

Software Research, Inc.
901 Minnesota Street
San Francisco, CA 94107 USA

Phone: +1 (415) 550-3020
TollFree: +1 (800) 942-SOFT [USA Only]
Fax: +1 (415) 550-3030
E-Mail: info@soft.com

Technology Builders, Inc. (TBI)

Technology Builders, Inc. (TBI)
400 Interstate North Parkway Suite 1090
Atlanta, GA 30339 USA
770-937-7900
Fax 770-937-7898
Info@tbi.com <mailto:Info@tbi.com>
www.tbi.com <http://www.tbi.com>

About TBI

Technology Builders, Inc. is an Atlanta-based software and services company providing integrated solutions for enterprise and e-business application development within the Automated Software Quality, Business Information Management and Internet and Client/Server markets. Through the TBI-Caliber=AE suite of software products and strategic vendor partnerships, the company provides best-of-class technologies in requirements management, requirements-based testing, test case design, configuration management, test planning and management, and defect tracking. Ranked 15th on the Deloitte & Touche Fast 500 list, TBI maintains a client base of over 500 companies, many among the Fortune 1000.

Caliber-RM

TBI markets and sells Caliber-RM=AE, a collaborative, Internet-based requirements management system that enables project teams to deliver higher quality applications. In an effort to detect and eliminate requirement errors and deficiencies earlier in the development cycle, Caliber-RM allows teams to fully define, manage and communicate changing requirements. Changes to requirement data such as traceability, status, priority and more are recorded and stored in Caliber-RM's central repository, providing reliable, up-to-date information for effective requirements-driven development and testing.

Caliber-RBT

Another TBI product, Caliber-RBT, formerly SoftTest, is a requirements-based functional test case design system that drives clarification of the application requirements and streamlines the testing process. Caliber-RBT enables project teams to analyze and refine requirements to eliminate ambiguities and conflicts, then use those requirements to design a minimum set of test cases for functionally complete test coverage.

TESCOM

TesCom is the leading international provider of software testing and quality management services, with over 700 consultants operating in Europe, USA and the Far East. During our eleven years of operation, we have developed effective testing solutions based on our wealth of experience across a broad range of sectors and technologies such as: load/performance testing, test automation, e-lab test centre, Interactive Digital Television (iDTV), WAP, e-security, usability testing, and network testing. For more information visit www.e-testing.com.

Our main service offerings include the following:

- Test strategy
- Test planning and design
- Test execution
- Test automation
- Test Centres
- Training on testing standards, methods and techniques
- Test management
- Project audits & reviews
- Risk assessment and management
- Testing benchmarks
- Tools selection and implementation
- Configuration management

Our experience covers a broad range of technologies and application areas, including:

- E-Business systems
- ERP/ERM systems
- Security systems
- Financial products
- Real-time systems
- Security systems
- Biotechnology systems
- Internet and new media systems
- Customer relationship management systems
- Euro systems
- Billing systems
- Embedded systems
- Healthcare systems
- Communication systems

TESCOM

TesCom UK Ltd.
5 St. John's Lane London EC1M 4BH
T: 020 7250 4705 F: 020 7250 0464

UPSPRING Software™, Inc.
15 Third Avenue
Burlington, MA 01803
Tel: 781-359-3300 or 1-888-9-DISCOVER
Fax: 781-359-3399
Web: www.upspringsoftware.com
E-mail: info@ upspringsoftware.com

UPSPRING Software, Inc. (UPSPRING) is a software infrastructure solutions company that develops, markets, and supports two compatible product lines for software development; CodeRover™, a targeted family of desktop software infrastructure applications to enhance personal productivity of individual software developers, and DISCOVER®, an enterprise-wide solution for organizational productivity.

CodeRover applications come in three categories; Productivity, Quality and Change, with many individually purchasable applications. CodeRover applications are downloadable over the Web, are easily self-installed, and are fully supported through Internet and telephone hotlines. Most CodeRover applications include the CodeRover Browser and are available with optional annual maintenance. Maintenance automatically provides upgrades to new versions as they are released. CodeRover applications are compatible with the complete enterprise-wide DISCOVER software Development Information System.

UPSPRING's flagship product, DISCOVER, is a complete enterprise-wide, software Development Information System made up of a series of integrated solutions created to address some of today's most challenging software development problems. By using DISCOVER, enterprises can rapidly evolve the software infrastructure of their existing eBusiness and other mission critical software applications. DISCOVER analyzes source code and related artefacts, creating a Web accessible, scalable, common database of information (the Information Model). The Information Model captures the relationships between all entities resulting in a high-level architectural perspective and a detailed view of the entire application, which can be shared across the organization and monitored over time.

DISCOVER enables software professionals to more thoroughly understand Their software systems, to more efficiently and accurately effect changes to a large body of source code, and to more easily reengineer or reorganize a complex software system, thus improving organizational productivity and quality, while reducing cost and time-to-market. All DISCOVER solutions come with turnkey services to assure effective deployment and ongoing customer success.

TESTFRAME

Getting testing under control

Organizations operate in a turbulent market. More and more the need is felt to prevent interruptions of core business processes by computer failure. CMG has gained a wealth of experience in setting up test organizations using their full scale method for structured testing, named TestFrame. The emphasis in TestFrame is on the strategic support of the complete testing processes, from organization to test execution, in which all test products are put in place and related to one another. The underlying concept of TestFrame is that a test is set up right from the start with maintenance in mind, so that future checks can be carried out with the minimum adjustment to the test material. Because the testing products are re-usable, you will have recouped your investment after just a few re-tests.
For more information please contact the TestFrame Research Centre of CMG, telephone +31 348 45 40 00, fax +31 348 45 40 13, e-mail testframe@cmg.nl, Internet: www.testframe.com.

CMG
Information Technology

Organizations are stronger with CMG.

CMG FINANCE | CMG TRADE, TRANSPORT & INDUSTRY | CMG PUBLIC SECTOR | CMG TELECOMMUNICATIONS & UTILITIES | CMG BUSINESS SERVICES

gitek®

interaction through software

Gitek nv

Gitek nv was founded in 1986 and is in addition to the design and development of software, specialised in software testing. Gitek employs more than 120 people, with 40 professional software Test Engineers. Its customers include the pharmaceutical industry, the financial industry, the telecom industry and insurance companies.



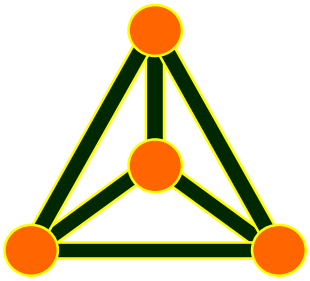
The difference

Gitek is an enthusiastic team of professionals, specialised in customised IT solutions and structured testing. A personal approach forms the basis of our success. This ensures better-adapted services to the customer's specific requirements, and a committed and motivated team ensuring optimal efficiency.

Structured Testing

Gitek is exclusive distributor of TMap® (Test Management Approach), TPI® (Test process Improvement) and TAKT© (Knowledge of Testing, Automation and Tools) in Belgium.
Gitek provides services that offer a complete solution for testing:

- Participation in the operational test process
  - Test planning and test management
  - Test design and test execution
- Complete test projects and fixed price projects
- Test advice and support
- Defining and implementing a structured test process
- Selection and implementation of test tools
- Improvement of the test process
- Training and coaching in testing



Contact us

Gitek nv
Sint Pietersvliet 3
B-2000 Antwerp (Belgium)
http://www.gitek.be
e-mail: gitek@gitek.be

elementool

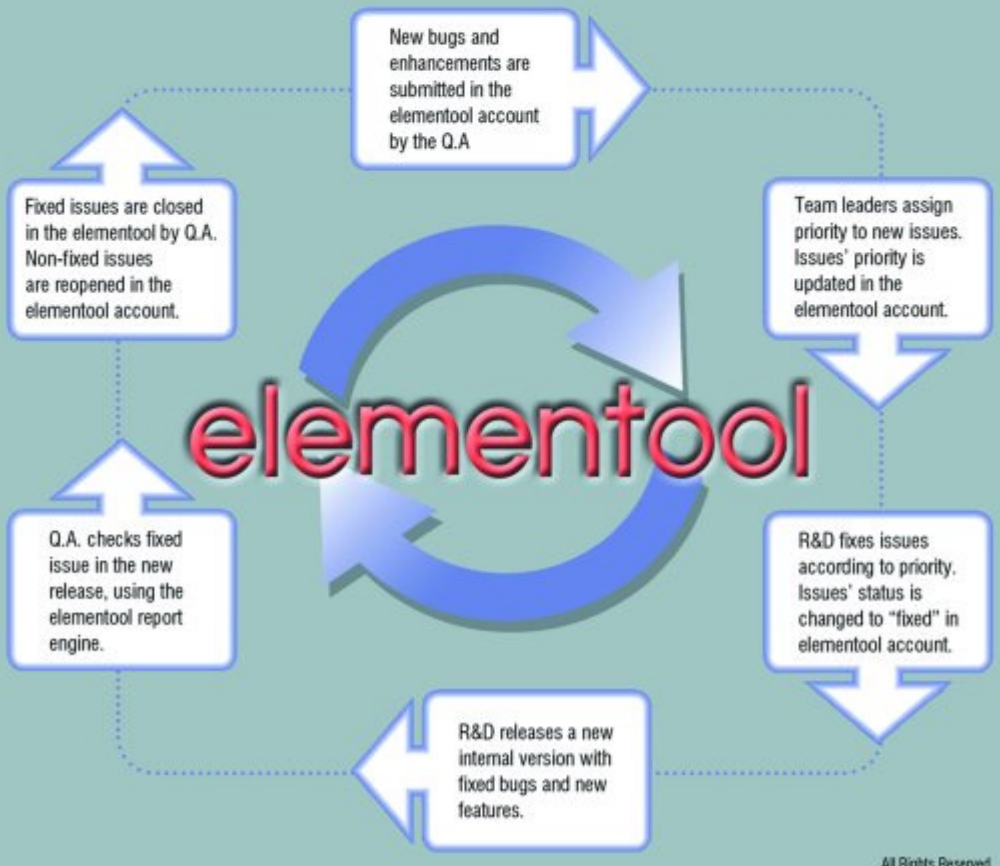
elementool, Inc.

http:// www.elementool.com / info@elementool.com

Benefits of elementool

Feature	
Free Service	Yes
Web based	Yes
Requires instalation of software	No
Customizable forms	Yes
Assigning Issues to team members	Yes
Powerful reports	Yes
Submitting Issues directly form your website	Yes
Mail notification	Yes
Self-download of your Issue list backup	Yes

Bug Life Cycle Model

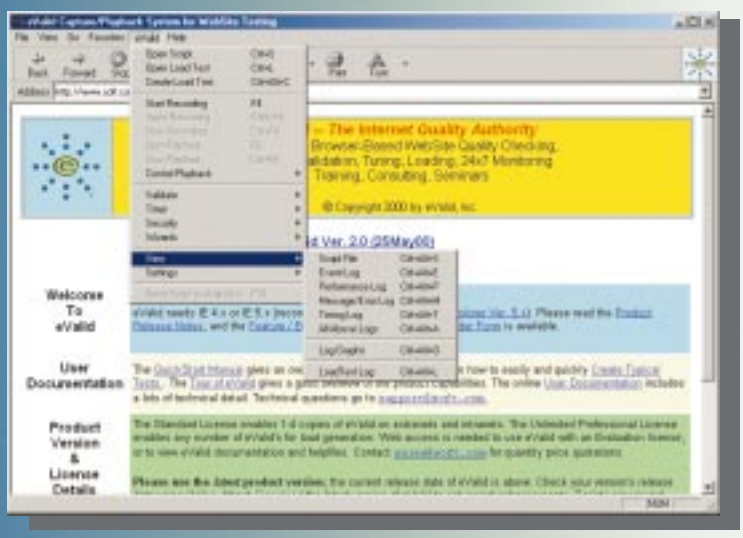


Quality Week Europe 2000

Internet Access Station Sponsored by

Leading Web-based Bug Tracking Tool
http://www.elementool.com

Quality Testing of WebSites



Make Your WebSite 100% reliable – automatically

- Recording
- Playback
- Script Creation
- Content Validation
- Performance Tuning
- Load Testing

eValid

Your e-Business Partner

www.e-valid.com