3

September, 2008

testing experience

The Magazine for Professional Testers

Test Techniques in practice -

Do they help?
Why do we often test without them?

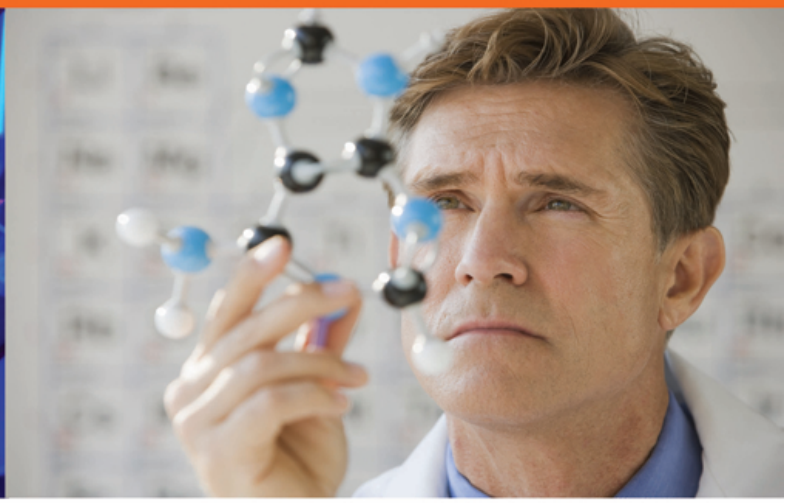
printed in Germany

free exemplar

www.testingexperience.com

ISSN 1866-5705

SELA COLLEGE Training for Professional Career Tracks in QTP



Sela offers a wide array of **QTP** and environment-focused automation training world-wide. The courses include both theoretical and practical hands-on knowledge sessions to bring the participant to the desired level of implementation under the supervision of certified and experienced instructors.

The courses are systematically structured as **Professional-Career Tracks** and modular packages or customizable Corporate Training Solutions streamlined with your company-wide technology needs, wherein you can choose/customize corporate courses in **QTP** based on your companyís technology environment and needs.

Courses

- QTP0100 - QTP First Step Windows
- QTP0101 - Scripting QTP Basic
- QTP0102 - Scripting QTP Advanced
- QTP0103 - Mastering QTP
- QTP0200 - QTP First Step Web
- QTP0201 - QTP Mastering the Web
- QTP0202 - QTP Mastering the Web Extended
- QTP0203 - QTP Mastering Web Services
- QTP0300 - QTP Mastering .NET Basic
- QTP0301 - QTP Mastering .NET Extended
- QTP0400 - QTP XML Data Suite
- QTP0401 - QTP ADO Data Suite
- QTP0500 - QTP Testing Multi-language Applications
- QTP0600 - QTP Mastering Java Basic
- QTP0700 - QTP Terminal Emulation Suite
- QTP0800 - QTP SAP for Windows
- QTP0801 - QTP SAP for Web

Professional-Career Tracks

- Discovering QTP
- QTP For .NET and .NET-Web Programmer
- QTP For Web Programmer
- QTP For Data Oriented - All QTP Programmers
- QTP For SAP and SAP- Web Programmer
- QTP For Multi-Language - All QTP Programmers
- QTP For Terminal Emulation Programmer
- QTP For Java Programmer Track

Sela offers you also other courses:
Certified Tester Foundation Level ▪ ISTQB - Certified Tester Advanced Level - Test Analyst ▪ ISTQB - Certified Tester Advanced Level - Test Manager ▪ Agile/Scrum/Certified scrum master course and much more.

Sela Israel: Phone: +972-3-6176066 ▪ Fax: +972-3-6176605 ▪ www.sela.co.il/en
Sela Canada: Toll Free: 1-800-640-4754 ▪ Fax: +1-416-628-3801 ▪ www.selacanada.ca
Sela India: Phone: +91-20-6521-6466 ▪ Fax: +91-20-2605-3565 ▪ www.sela.co.in

Editorial

I hope you have enjoyed the summer time and spent some days at the beach or the mountains. I couldn't. I really would love to spend some days at the beaches of Gran Canaria, just laying down and having a nice cold beer and fish or a paella at the "Las Canteras"-beach promenade for dinner … But that's life! I had to work.

In this issue we focus on test techniques. Test techniques are the key to success. It doesn't matter if you use them systematically or just based on your experience performing exploratory testing. We are lost without these techniques and experiences. Some of us use regularly equivalence partitioning, boundary check, error guessing, decision tables and paiwise for our test and perform them systematically. Some of us don't.
If you have a look on the market there are a lot of books and seminars which explain these techniques. My impression based on my old days as tester and actually as manager of a "testing company" is that most people involved in this field don't use to apply them for the daily work. Why?

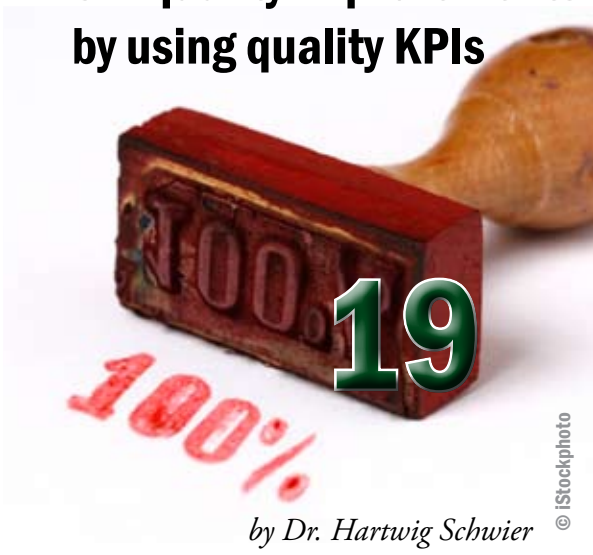
My personal opinion is that the market is leaded by tools companies that don't use them directly. If you have a look they support test management, test processes, test automation but not really test techniques. Test techniques as the key for the success may have been forgotten for a long time.

In this new issue there are some articles that give you tips for the leisure time, for test case design, test management and share experiences or new ideas. Please enjoy them.
We have started a serial of interviews to presenting interesting people worldwide and their work, thoughts and strategies that influence our field.
With the second issue we have achieved over 80,000 readers! I want to thank all of you for the support and for spreading the magazine in your community.
We are working hard on a new website with community character that should appear in a few weeks. We will start with a beta version and hope that all of you find a lot of bugs!
I wish you a pleasant time with testing experience.

Yours sincerely

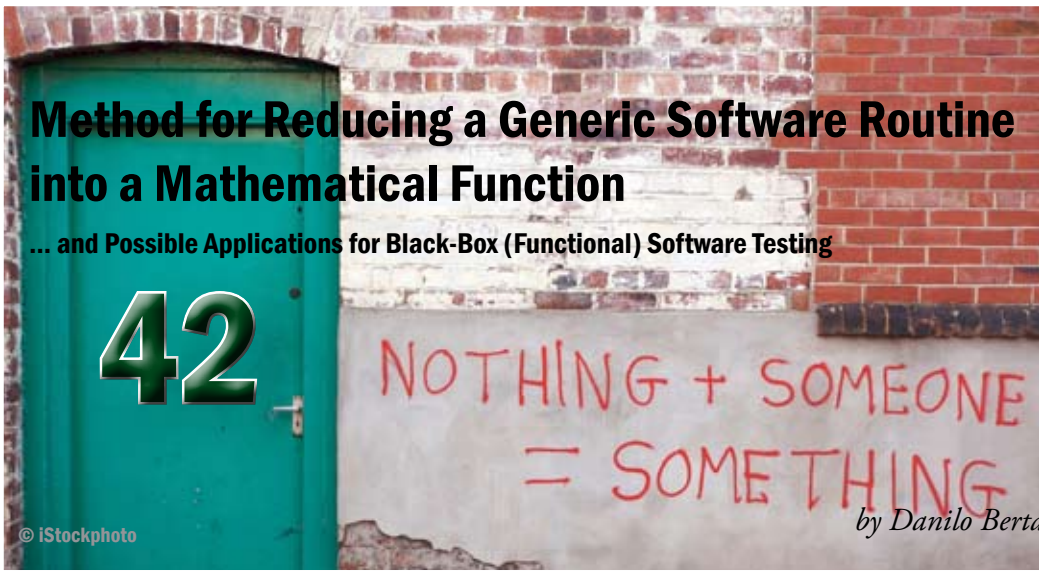
José Manuel Díaz Delgado

Contents



by Dr. Hartwig Schwier

testing experience



Method for Reducing a Generic Software Routine into a Mathematical Function

... and Possible Applications for Black-Box (Functional) Software Testing

42

by Danilo Berta

The benefits of modeling in a large-scale test integration project: A case study

by Graham Bath

25

European Tour

Application Security Testing

2 days testing course by Manu Cohen-Yashar

Limited places

3rd - 4th November Berlin
6th - 7th November Amsterdam
1st - 2nd December London
4th - 5th December Stockholm

We are living in a world of data and communication, in which the most valuable asset is information. There is no doubt that today's applications must be secure.

Security Standards are created to insure that products will implement security measures to protect their data.

Security is an "all-inclusive" term, which means it must be implemented "everywhere", in all levels:

- **Users:** Train your users and build awareness to help them to reduce the risk of performing irresponsible actions which will be used by the attacker. Make sure your UI helps your user to take the correct decisions.
- **Infrastructure:** Firewalls, Network Admin, Host & Server Hardening, Network traffic encryption etc.
- **Application:** Authentication, Authorization, Input validation, Encryption, Configuration management, Parameters manipulation, Auditing, Error Handling etc.

The application must be designed and implemented while taking security issues into consideration. We have to remember that the attacker needs to find just one security breach while we have to protect everywhere.

Leaving one of the above levels unhandled will result in a completely unsecured product.

Application security is not just another feature. You can not just turn it on.

Application security demands a lot of thinking. Threat modeling and a lot of design work must be done. Many concrete actions must follow in every phase of the development cycle.

Security Testing:
Testing is a crucial part of Security Development Lifecycle.
The tester must understand methodology of secure development. He has to build a security test plan using the threat modeling documentation.

The tester has to understand the Hacking mechanics. He has to get out of the box and think like a hacker.
The tester has to know the security testing methodology. The hacker must be diligent and work systematically to find security breaches.

All this and more will be taught in the course.

1250,- EUR
(plus VAT)

testing experience
Knowledge Transfer

Please register by
email kt@testingexperience.com
or fax +49 30 74 76 28 99

News

Improve Quality Services and iSQI launch IREB Requirements Certification in The Netherlands and Belgium

Requirements engineering is a key discipline for ensuring project success. The function of requirements engineering during the development process is to determine the system requirements, to document them adequately, to test them and to manage them throughout their entire life cycle. The requirements engineering discipline has recently been receiving much attention. Not at least of course as a result of all the outsourcing that is taken place.

The International Requirements Engineering Board

A number of years ago the International Requirements Engineering Board has been founded (www.certified-re.de) within the Requirements Engineering discipline. The stated aim of the International Requirements Engineering Board (IREB) eV is to help standardise basic and continuing training in requirements engineering by providing syllabuses and examinations. Ultimately, it seeks to improve actual practices in requirements engineering. The members of the Board are independent, internationally recognized experts in economics, consulting, research and science and include Chriss Rupp (chair) and Suzanne Robertson. IREB is a registered non-profit association.

Improve Quality Services

Improve Quality Services is the leading course provider in the areas of testing and quality management in The Netherlands and Belgium. For a number of year Improve Quality Services has already been running Requirement Engineering and Management courses. They have recently extended their services to also provide the "Certified Professional for Requirements Engineering Foundation level" course, compliant to the IREB requirements. The course is intended for anyone who works with requirements in their professional lives. It covers the fundamentals of requirements engineering with regard to determining, analysing, specifying, documenting, testing, and managing requirements.

International Software Quality Institute (iSQI)

The certificate is awarded to anyone who passes the exam which is prepared by the IREB. The certificate is an impartial confirmation of the level of knowledge the holder has attained during his training in requirements engineering. The international exam institute iSQI is responsible the examination (www.isqi.org). The iSQI certifies IT personnel all over the world. It also amongst others runs examinations for ISTQB and QAMP.

More information

The IREB Requirements Certification Foundation courses in The Netherlands and Belgium will start this autumn. More information can obtained via the web site of Improve Quality Services: www.improveqs.nl, or by sending an email to info@improveqs.nl.

Stephan Goericke (iSQI) and Erik van Veenendaal (Improve Quality Services) signing the agreement.

The benefits of modeling in a large-scale test integration project: A case study

by Graham Bath

Abstract

This case study describes the benefits obtained and the lessons learned from introducing a model-based approach into one of europe's largest system integration testing projects, which includes off-shore components and a staff of more than 400.

An industrial-scale research project with three distinct phases was established to manage the test process modifications resulting from adopting a model-based approach and to ensure that practical and business factors remained in focus throughout. In this paper the three phases are described in terms of the concept used for achieving defined objectives, the lessons learned, and the actual benefits obtained.

The results of the case study show that the model-based approach is an effective instrument for enabling a wide range of improvements to be achieved. The customer requirements process has been integrated into the testing process and a seemless, tool-supported process from requirements through to test specification has been achieved.

1 Introduction: The trend to highly integrated systems

Monolithic systems are a dying breed. The trend these days is towards system architectures which are sometimes best described as "systems of systems". The reasons for this could fill a whole article, but certainly the benefits of software re-use, the availability of standard software packages, the flexibility and scalability offered by heterogenous architectures and the "mix and match" possibilities offered by web-based software services (SOA) are among the principal driving factors. But there's a price to be paid for developing architectures like this; the task of testing and providing quality-related information to decision makers is generally more complex.

It's against this background of "systems of systems" that model-based approaches was evaluated and introduced. This article describes a number of the benefits which were achieved and the strategy which was followed. Since the paper is based on industrial experience, it would come as no suprise to learn that there were problems along the way, so I'll be sharing some "lessons learned" as well.

2 Deciding on an approach: Somewhere over the rainbow...

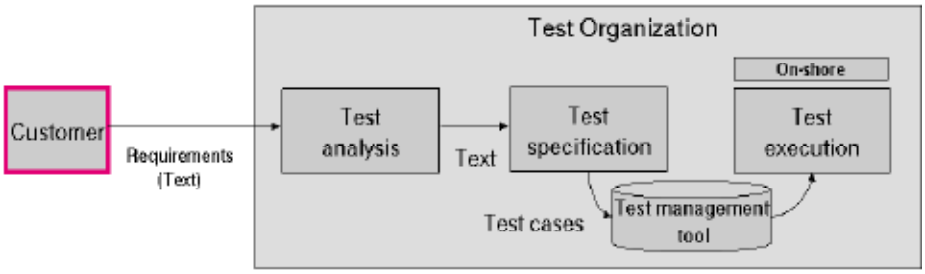
There's an old saying that if you can find the point where a rainbow ends you'll find a pot of gold. The problem is, if you set off towards your rainbow's end you're chasing a moving target, which may ultimately disappear before your very eyes. We never actually reach that mystical point where the pot of gold is buried. In IT we've all been there before haven't we? We've chased pots of gold called "tools give full test automation at the press of a button", "mainframe applications will be entirely replaced by e-commerce" and, more recently, "just send everything offshore; it's sure to be more efficient". We probably all know by now that there's more to it than that. Yes, tools can help automate certain types of testing efficiently. Yes, e-commerce is a major step forward for user-centric applications and, yes, offshore can yield substantial benefits if managed properly. But we've been around long enough to know that these new developments follow a cycle which starts with euphoria, passes through the "valley of dissolusionment" and finally reaches the a level of practical usage (which is generally somewhere between those two extremes).

At T-Systems, Test Factory we were acutely aware when we set up our study that using a model based approach for achieving our testing objectives could well end up like chasing the next pot of gold. So we need to be careful and not be too euphoric at the start and we adopted an iterative approach, where learn-ing from experience was a central element.

Oh, and we needed a realistic objective: "To achive synergies and efficiencies by introducing a model-based approach and integrating this into our established standard testing process; from planning through to execution."

2.1 The starting point

The existing test process is based on the description provided by the Interna-tional Software Testing Qualifications Board (ISTQB). It was not an objective to replace this test process, but we certainly wanted to make it operationally more efficient. The diagram below illustrates the aspects of the test process which were identified as candidates for efficiency improvements (note that this diagram does not show all of the steps in the test process).



The primary areas for potential improvement were identified as:

- the customer requirements interface
- the analysis and specification of test cases
- the use of off-shore organizational components

2.2 The concept

To investigate the benefits of using models in testing a research project was set up which works closely with both operational projects and the local university, (in this case the Technical University of Munich). The clear intention was to obtain buy-in from the projects right from the start, avoid the problems of developing new concepts in "ivory towers" and to make use of existing academic expertise in the field of modeling.

The project plan we developed had three phases, each with approximately six month's duration. The objectives for each phase were only set after learning from the previous phase.

The three phases we identified are shown in the table below:

Project phase	Name of phase
1	"Let's get modeling"
2	"Integrate, integrate"
3	"Automate, where it makes sense"

Details of the project phases will be discussed in later chapters. Before we started with phase 1 though, it was important to choose a suitable pilot project.

2.3 Choosing a pilot project

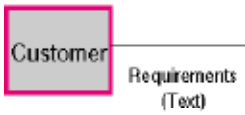
The objectives we set ourselves could only be achieved if a representative project was used as a pilot. Of course, it's possible to reduced risks by using a relatively simple project for piloting, but what value are the results afterwards? Will they scale to the types of large, complex project we have to test?

What we needed from our pilot project were a number of attributes so we could be sure the approach was going to provide benefits when rolled out. We selected a major testing project as our pilot because it gave us good coverage of these attributes.

Required attributes for pilot	Characteristics of the chosen project
Technical complexity: Large numbers of applications and interfaces.	Over 70 inteconnected applications
Business processes complexity: Several applications communicate with each other to implement business processes	Mostly complex business processes running over multiple applications. Many "end-to-end" tests are performed.
Organizational complexity: Geographically distributed organization, including off-shore elements	The project is performed at several locations in Germany and in India.
Real business case: There has to be a reason for doing this. We're not interested conducting a purely academic exercise.	With over 400 testing professionals involved in the project, any improvements to efficiency will lead to substantial savings.

2.4 The chosen pilot project

The fundamental task of the pilot project is to provide functional and operational acceptance testing of the entire system stack prior to entering production. Regular releases of software are tested and quality-related information provided on which production decisions are taken. For each release the testing organization receives a number of text-based "solution documents" from the business owners which are used as the test basis.



3 Phase 1: "Let's get modeling"

In this first phase we looked carefully at the issue of modeling in general and established an overall concept for introducing model-based approaches to our test process.

3.1 The overall concept

Central elements of the concept were:

- Use of UML 2.0 to model the test basis. In this step (which we called "design for test") the business requirements provided in the customer's solution documents were captured as UML2.0 Sequence Diagrams and Class Diagrams.
- Use of the UML-2 Testing Profile (U2TP) to take over the UML diagrams and extend them for test purposes. The result is a Test Model (equivalent to a test specification) which contains information about the System under Test (SUT) and each test case.
- Use the Test Model as a basis for test execution. This could be performed either manually or (potentially) used by tools for the generation of automatically executable tests.
- For each of these steps the roles and tasks for on-shore and off-shore staff are clearly identified.
- Identification of business processes for implemention in UML. These needed to be stable, well understood processes and the responsible test designers needed to be willing to take on the piloting task in a "live" project.
- Training in UML 2.0 provided by the Technical University of Munich. A total of 12 experienced staff were selected to perform the phase 1 modeling
- Selection of the Enterprise Achitect (AE) tool for the UML 2.0 modeling. The selection decision was based primarily on ease of use and low license costs.
- Modeling of 30 selected business processes as UML 2.0 activity diagrams.
- Modeling of the messaging and communications between individual systems using UML 2.0 sequence diagrams.
- Getting feedback from industry: The concept was presented at the International TTCN-3 conference in Berlin in 2006 and at the System Engineering Conference "SE08" in Munich in 2008.
- The basic concept for phase 1 is shown in the diagram below.

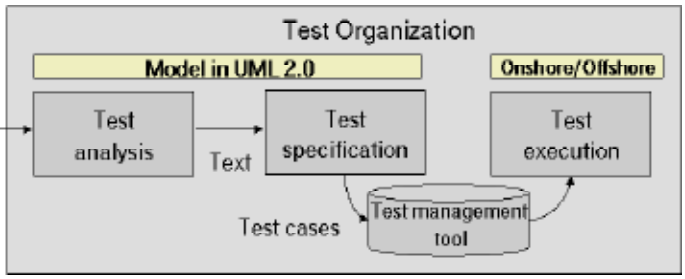
3.2 Strategies for off-shore compared

Before we could really settle on the concept outlined above we first needed to consider the issue of off-shoring, which is an essential element of our testing organization. What would be the best way to capture the customer's requirements as a test specification so that our off-shore teams could use it efficiently and effectively? We looked at three different options:

1. **Text-based**: This is the existing baseline. Textual test cases were generated onsite, translated along with other specification documents and then sent to offshore for execution.
2. **Model-based on-shore**: Test design and specification both conducted on-site and then executed off-shore.
3. **Model-based partial off-shore**: Test design conducted on-site. The design is then refined and the test specification created off-shore. After a review with the on-shore team the test are then executed off-shore.

The text-based approach (option 1 above) in which test cases are translated from German into English was known to be inefficient (that's one of the primary reasons for looking at modeling). Here are a few reasons why:

- Lower proportion of work carried out off-shore.
- High levels of communication due to queries regarding the test



specifications and incorrectly raised defect reports.
- Lower than expected levels of productivity.
- Insufficient knowledge transfer.
- The translations were often a major source of error and off-shore teams often found themselves (incorrectly) interpreting their meaning.
- Insufficient knowledge transfer.
- The off-shore staff were not involved in the testing process early on, which lowered motivation.

3.3 Benefits of the model-based approaches

A structured questionnaire was used to support the results obtained from the first phase. Even though this evaluation was primarily subjective in nature, it enabled all those participating in the innovation project to express their views on the benefits obtained. Did they "feel" that there had been improvements made? Could they perform their work better than before? Was the effectivenss of the testing now better or worse than before? Questions like these were important for giving us the confidence for taking further steps.

Here are the primary benefits which emerged from the survey:

- **Increased efficiency**. In some areas, efficiencies of over 50% were achieved compared to a testing approach without modeling. These efficiencies arose from a number of different sources (see below).
- **Better communications**. Perhaps one of the most significant benefits was in better communications within distributed testing teams. Testing staff in Germany and India could communicate via UML diagrams instead of pure text. There were fewer misunderstandings; fewer telephone conferences and a general improvement in the process of off-shore governance.
- **Higher motivation**. Levels of motivation for off-shore staff increased because they became far more involved in the test process as a whole.
- **Better knowledge capture**. The capture of requirements documents as models relieved the problem of "tacit" knowledge where particular individuals are in possession of key business process know-how, which is often not fully documented.
- **Stimulus for offshore**. In general a model-based approach proved to be a significant "enabler" for the off-shore testing process. Whilst many papers and presentations have highlighted the automation-based benefits of a model-based approach to testing, the benefits we obtained pointed to significant improvements in other aspects of the software development process such as in knowledge management, team motivation and off-shore governance.

3.4 Lessons Learned from Phase 1

Not everything worked out absolutely as expected in this first phase. In particular, the following problems arose and a number a lessons were learned:

- The modelling task was actually more time-intensive than predicted, despite the training courses and the availability of expert support. We realized quite quickly that modelling guidelines are absolutely essential, especially regarding the levels of abstraction required.
- The use of Enterprise Architect (EA) as a modelling tool proved to be a double-edged sword. EA is easy to learn and use and allows the modeller complete freedom to capture requirements as models. This freedom, however, can cause problems, even when modelling guidelines are available. First of all, the modeller has to keep referring to those guideleines and, secondly, control mechanisms have to be in place to ensure that they are being correctly applied. One particular problem area was fixing the level of abstraction required in the model. This started to erode some of the efficiency gains.
- The levels of defects detected using model based test specifications with off-shore testing teams remained relatively constant.

3.5 Just a minute: why is the test organization doing this?

Phase 1 of the project was initiated in order to improve the efficiency of our testing organization's process. One of the principal tasks performed to achieve this was arguably, however, not a testing task at all. The modeling of requirements in UML was a necessary step to enable other benefits to be gained for the testing organization. In principal it's a desirable skill for a testing organization to be able to model requirements, but we were still left at the end of phase 1 with this basic testing overhead.

4 Phase 2: "Integrate, integrate"

During phase 1 of the project we became aware of similar kinds of project being conducted by our customer. Their idea was to model their requirements so that development organizations could use them. For us this was welcome news; it would be a relatively small conceptual step to also include the testing organization in their plans.

The contours of the next phase in the project began to take shape when we learned that a modelling tool had already been selected by the customer. The tool, MID-Innovator, would not only provided the "bridge" between our customer and the testing organization, it would also help to strengthen our modeling process and make it more precise compared to our earlier approach of using Enterprise Architect with modeling guidelines.

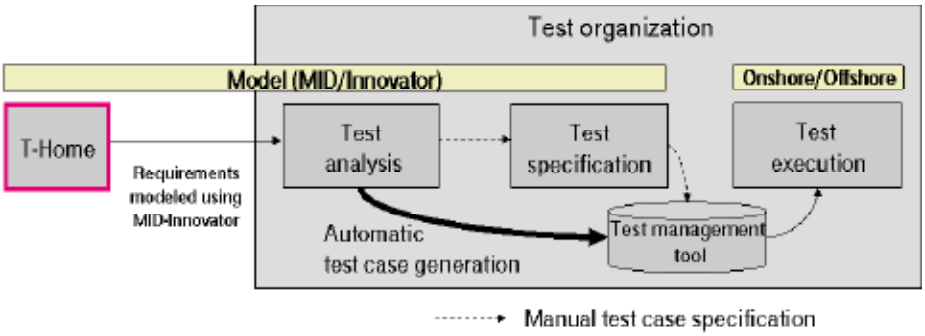
After discussions with our customer the potential benefits of project phase 2 were clear; if we could integrate the MID-Innovator modelling tool into our testing process we would have a seemless, model-based interface with our customer. The test organization would no longer need to model the customer's text-based requirements and significant gains in efficiency could be achieved. In addition, the use of this particular tool held the prospect of being able to automatically generate test cases from the models.

4.1 Concept for Phase 2

Introducing a new tool like MID-Innovator into the testing process triggered a number of integration tasks (hence the name given to this phase!):

- Integrate test-specific aspects into the requirements specification process. To do this we would implement a Domain Specific Language (DSL) so that both customer and test organization could agree on a common domain-specific language for modelling. The test organization would assist in creating this DSL so that our standard test process could be taken into account.
- Integrate the MID-Innovator tool into our own tool set. This would mean creating a new XML-based interface between MID-Innovator and our standard test management tool.
- Integrate the modelling tool into our existing testing process. This would mean altering a number of steps in the process, including test planning and test specification.
- Integrate the modified test process into off-shore components of our organization
- Establish a knowledge management concept which would ensure that best practices and lessons learned from the innovation project could be shared across the entire testing organization. To achieve this a Wiki would be developed using the Confluence product.
- Model the same test scenarios used in phase 1 to allow comparisons to be made.

The basic concept and objectives for phase 2 are summarized in the diagram below:



Note that the option to generate test cases automatically from the MID-Innovator-based models does not mean that the option to define test cases manually, using UML (as in phase 1) or indeed with any other valuable approach is eliminated.

4.2 Benefits and Objectives of Phase 2

At the time of writing this paper the second phase of the innovation project was under way. The benefits we expect from this phase include:

- Efficiency gains resulting from the use of a common DSL
- Efficiency gains resulting from the use of a common modelling tool
- Fewer defects found as a result of requirements problems
- More efficient generation and administration of test cases
- Easier monitoring and verification of achieved test coverage levels
- Better sharing of knowledge across the testing organization

In addition to the benefits noted above, we want to provide answers to a number of questions:

- How easy is it to create a common DSL and what are the primary areas to pay particular attention to.
- What proportion of the lessons learned are applicable specifically to the pilot project itself and what can be defined generically so that other projects can benefit?
- Will the modelling tool be able to generate a high-value set of test cases based on risk, or will expected efficiencies be eroded by an "explosion" of automatically generated test cases.
- Does our modified testing process handle requirements changes efficiently?
- How does this approach compare to the results of phase 1, where models were created in UML using Enterprise Architect.
- Is our customer satisfied with the seamless model-based approach?

The results of phase 2 will be documented as a business case which will enable management to decide on a general roll-out and the objectives of phase 3.

5 Phase 3: "Automate where it makes sense"

Assuming that we are satisfied with the results from phase 2, the next phase planned will look at automation. As the name for this phase suggests, we will be performing evaluations of automation concepts which integrate to our testing process.
Modular test automation concepts will be evaluated. These may be based on keywords defined in the DSL "meta-model", or may make use of existing concepts. Where a clear business case can be made for automation this will be implemented, potentially using our off-shore organization.
If we have been able to a well-defined DSL in phase 2, the opportunity may also exist for using tool-support to generate automatically executable test cases. There are already some products available to do this, but for the moment we'd like to let those products mature a little.
One of the principal objectives to be answered in this phase will be to define modeling frameworks at a high enough level of abstraction to make the benefits generally available in other projects and for other customers.

6 Summary and conclusion

We're experienced enough to know that "magic solutions" rarely become reality and that the day is unlikely to come where we can represent requirements in model form, press button "B" and then watch all the tests run automatically. Recalling the rainbow anecdote at the start of this paper, we're a long way off from this particular "pot of gold". However, we have shown in this industrial scale pilot project that some major benefits can be obtained from modeling in a complex large-scale integration project.
Here are the main benefits in summary.

- Modeling enables the benefits of off-shoring to be realized.
- Models can build bridges between customer and test organization, especially if Domain Specific Languages and common modeling tools can be utilized. Major efficiency gains can be achieved as a result.
- Modeling is an effective instrument in achieving effective knowledge management.
- Modeling can help introduce rigour into testing, especially if tools can be properly integrated into the test process.

I look forward to informing you of progress. Who knows, we may find the occasional "pot of gold" along the way!

Biography

Graham's experience in testing spans over 25 years and has covered a wide range of domains and technologies. As a test manager, he has been responsible for the testing of mission-critical systems in spaceflight, telecommunications and police incident-control. Graham has designed tests to the highest level of rigor within real-time aeropsace systems such as the Tornado and Eurofighter military aircraft. As a principal consultant for the T-Systems Test Factory he has mastered the Quality Improvement Programs of several major German companies, primarily in the financial and government sectors. In his current position, Graham is responsible for the training programme and for introducing innovative testing solutions to his company's large staff of testing professionals.

Graham is a member of the authoring team for the new ISTQB Advanced Level Certified Tester and is a long-standing member the German Testing Board, where he chairs the Advanced Level and Expert Level Working Parties.

Together with Judy McKay, Graham has co-authored the recently published book "The Software Test Engineer's Handbook", which is a study guide for the ISTQB Test Analyst and Technical Test Analyst Advanced Level certificates.

© iStockphoto

SW quality improvements by using quality KPIs

by Dr. Hartwig Schwier

1. Introduction

Océ supplies digital printing systems, software and services for the production, reproduction, distribution and management of documents in small and wide format for professional users in offices, educational institutions, industry, advertising and the graphics arts market.

To make the print workflow faster and more efficient R&D Application Software within Océ develops and maintains modular document workflow software. These products are commercial-off-the-shelf (COTS) software products and licensed to customers. The base products can be configured by options and additional components can be added to support integration with system and work processes already in place across the full range of printing environments.

In 6 R&D sites worldwide approx. 220 engineers develop and test about 10 product lines with independent life cycles.

At the beginning of 2007 software quality key performance indicators (KPI) on product level were introduced

- to report the quality of the software products
- to initiate software product quality improvement activities

In addition, all quality product indicators contribute to a global Q-indicator, which represents the overall product quality of R&D Application SW.

The defined quality indicators reflect the actual "quality in use" of the software products from a customer point of view. Besides the number of defects and change requests, trends, priorities or backlogs of the open field reports are also considered.

2. Definitions

2.1.1.1. Problem reports

As the quality indicators should reflect the actual "quality in use" of the software products from a customer perspective, the evaluation of the metrics is based on a corporate-wide defect database, which contains all customer and service records worldwide of released products. Each database record includes priorities, time stamps, descriptions, responsibilities and the actual status for the problem solving process.

2.1.1.2. Product and product groups

As problems are reported in relation to products, options, components or configurations, it is necessary to define exactly what is understood by product or product group.
In addition, to define a global quality indicator, the relative importance for each product line across business segments was derived from the number of installations (software licenses reports) and the impact on the overall business. The product weights were set as 1, 3, 5 and 9. A weight of "one" is meant as "nice to have / limited importance" and a weight of "nine" stands for "business crucial".

2.1.2. Standard SW quality metrics

Despite the fact that the usage of metrics and KPIs is highly recommended, real practical examples for software KPIs and their application can seldom be found in literature. Therefore, we analyzed state-of-the art metrics definitions (see references listed in the appendix), selected the most promising for our purposes and tailored them for the Océ R&D Application Software processes to cover all relevant quality aspects, e.g. number of problems, trends, severity of problems and time-to-solve (backlog). Product-specific parameters allow making the definitions applicable to the variety of all 10 products groups.

The relative weight for each metric was derived systematically from experience and observations, e.g. starting from initial statements like "too many open problems" or "response time to solve is too long".

In table 1 the defined metrics, the description and the relative weight are summarized.

Metric	Weight	Description
TCP	9	**T**rend of **C**ustomer **P**roblems measures the deviation from reference level. Defined as unilateral capability index and displayed as pseudo control chart.
PCI	9	**P**roblems per **C**ustomer **I**nstallation
ASCP	3	**A**verage **S**everity of **C**ustomer **P**roblems
Backlog	5	**Backlog** of open problems (response-time-to-solve)
CRI	1	**C**hange **R**equests per Customer **I**nstallations
NCR	0	**N**umber of **C**hange **R**equests (normalized)
CRISIS	0	Number of **Crises** (absolute measure)

Table 1: Defined metrics, their description and the their relative weight

2.1.3. Definition of Quality indicator

The q-indicator for a product as well as the global quality indicator are defined either as weighted sum of the metrics or as weighted sum of the product q-indicators.

For the product q-indicator the definition is as follows

$$q = \frac{\sum_i w_{mi} * m_i}{\sum_i w_{mi}}$$

m_i Metric
w_{mi} Weight for metric
i Metric identifier

and the global Q-indicator as

$$Q = \frac{\sum_i w_i * q_i}{\sum_j w_i}$$

q_i q-indicator per product group
w_i Weight for product group
i Product group identifier

The interpretation of the quality indicators is identical and straight forward (see table 2 for details)

Value ranges	Interpretation and color assignment
0 ≤ m, q, Q ≤ 30	Quality level is good
30 < m, q, Q ≤ 60	Quality level is acceptable, but to be improved
60 < m, q, Q ≤ 100	Quality level poor (to be corrected)

Table 2: Interpretation of the metrics and quality indicators

2.1.4. Reporting

The indicators are reported on a monthly basis. Figure 1 shows the typical cover page of such a report, which summarizes all important findings per product in a radar diagram. The smaller the size of the area defined by the blue product line, the higher the quality of the product.
The status of a product group is displayed also in the pie-chart; the size of the pie represents the impact on the overall Q-Indicator.

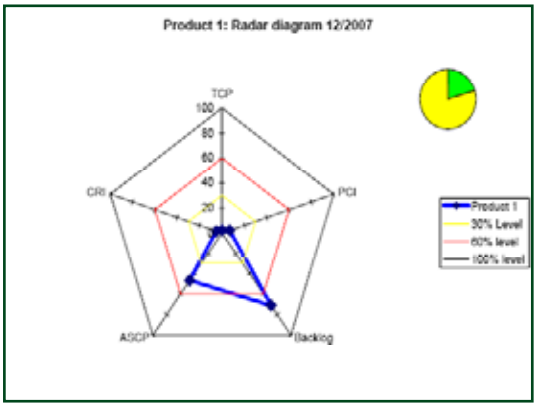


Figure 1: Product quality KPI displayed as radar diagram

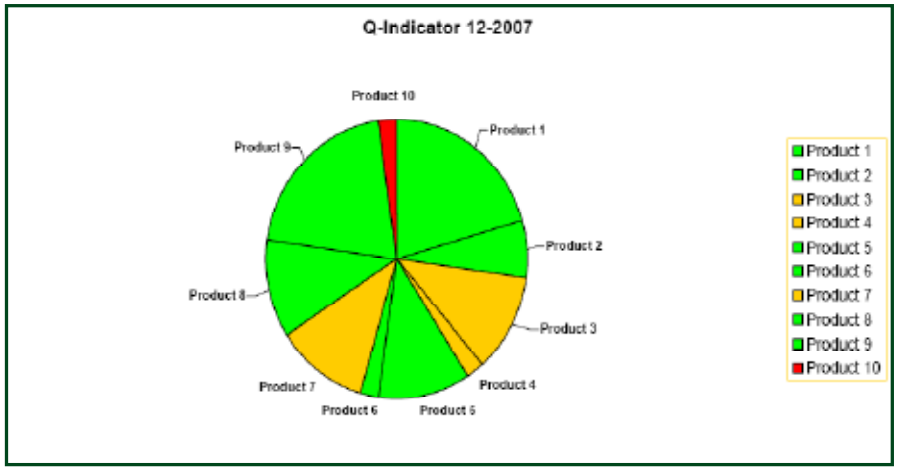


Figure 2: Graphical presentation of the Global Q-Indicator (27.48, green) of December 2007 as a pie-chart.

The Q-indicator is reported monthly as red (poor), yellow (acceptable) and green (good), but also in more detail in a percent range from 0 (very good) to 100 (very poor). In Figure 2 the global Q-KPI (27.48, green) of December 2007 is shown as a pie-chart. The color of each pie displays the status of a product group and the size of each pie represents the impact on the global Q-indicator respectively.

3. Findings and Improvement Results

3.1. Findings

After the introduction at the beginning of 2007, a detailed analysis showed that the number of problems was too high, that there was a strong deviation from the reference values, that the average problem priority was too high and the problem fixing time too long. The following root causes were identified:

- Inconsistencies in the problem database and in the problem handling process, e.g. no systematic record handling or limited attention on record maintenance.
- In the planning of product releases, the attention was more directed on new functionality than on product quality improvements.

The agreed actions were straight forward:

- Purge the database with the support of service and development.
- Initiate and define product quality improvements explicitly either as part of the regular functional enhancements or as separate releases.

3.2. Improvement results

3.2.1. Global Q-indicator

By December 2007, the overall Q-indicator had improved by about 40% from 45 (yellow) to 27 (close to green). Figure 1 shows the result of December 2007 as a pie-chart and Figure 3 displays the improvements achieved in 12 months.

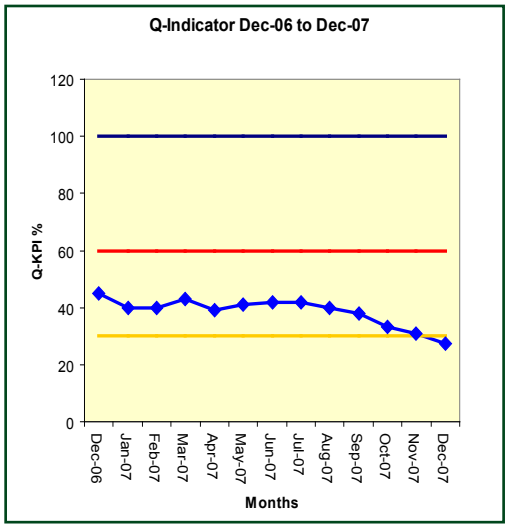


Figure 3: Achieved improvements of the overall Q-Indicator in the year 2007.

3.3. Product quality indicator

Figures 4, 5 and 6 show the significant improvements over the period of about 12 months for one product.
The green polygon in Figure 4 describing the quality in 2007 is almost completely included in the 30% level (yellow line). In addition, the area defined by the green polygon is only a fraction of the area defined by the red polygon (December 2007) indicating the degree of quality improvement.

The reduction of number of problem records of about 66% could be achieved as a combined action of four (!) bug-fix releases and additional maintenance activities of purging the problem database (see figure 5)

The columns in Figure 6 represent the number of open problems per priority. Besides a significant reduction in open problems (as discussed before) the distribution of the data of December 2007 in green can obviously be fitted as a Gaussian distribution with a mean value close to priority 3. From a customer point of view, this means to our experience a high product quality in functionality, performance and usability.

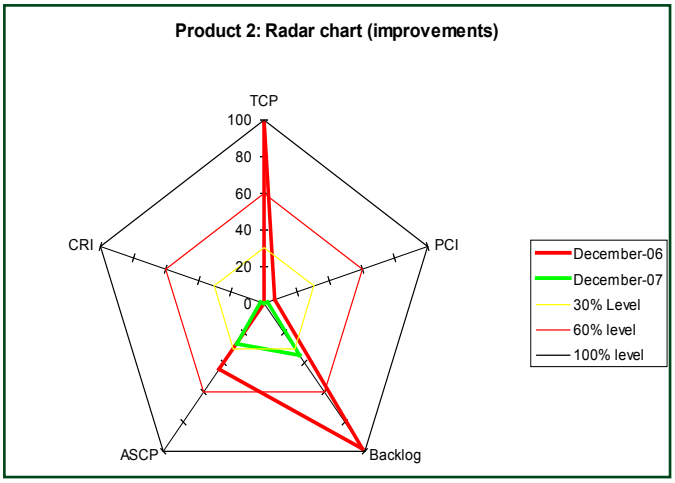


Figure 4: Radar chart indicating the improvements

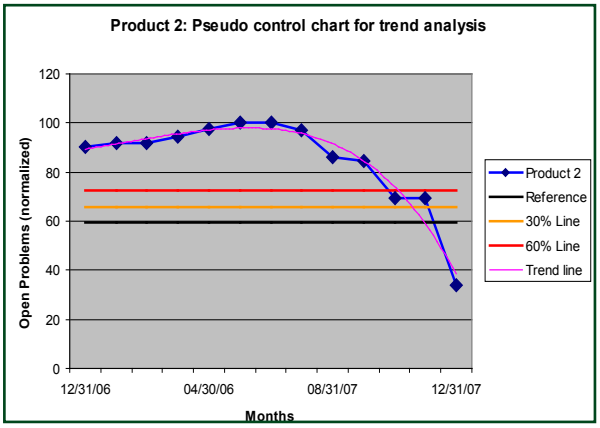


Figure 5: Decrease in the number of problems for one major product.

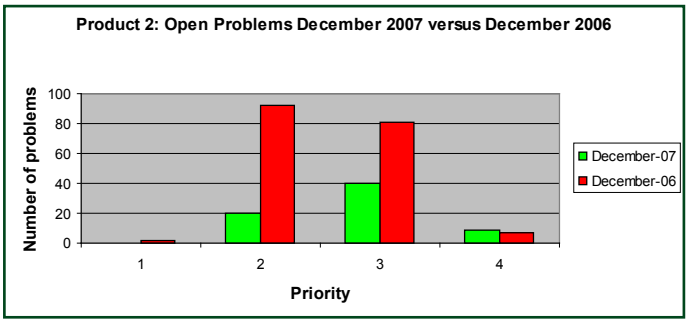


Figure 6: Open Problems and average problem severity

4. Summary

During 2007 software quality indicators were introduced in a systematic way. The defined KPIs describe several aspects of product quality, e.g. the number of problems, the severity of problems and the problem handling. It was proven that quality indicators can be used for reporting as well as for initiating and controlling improvement activities.

To be successful a systematic approach, regular reporting and the support from all departments involved is needed.

5. Appendix: Literature

The introduced metrics are based on state-of-the art definitions and adapted to the available data and defined processes at R&D Application Software.

The following sources were used:

- Kan, H. - *Metrics and models in software quality engineering* 2nd edition, Addison Wesley, 2003
- Galin, Daniel - *Software Quality Assurance* Addison Wesley, 2004
- Craig, Rick D; Jaskiel, Stefan P. - *Systematic Software Testing* Artech House, 2002

Biography

Hartwig Schwier is head of corporate software test and validation at Océ. He has been working in software engineering for more than 20 years covering software development, architecture and quality assurance. In his recent position he is responsible for software testing at six R&D sites worldwide and for the time being he is also general manager of Océ Software Romania in Timisoara.

Are test design techniques useful or not?

by Hans Schaefer

Many people test, but few people use the well-known black-box and white-box test design techniques. The technique most used, however, seems to be testing randomly chosen valid values, followed by error guessing, exploratory testing and the like. Could it be that the more systematic test design techniques are not worth using?

I do not think so. When safety-critical software is produced, it is tested very systematically using the good well-known techniques. Standards recommend or require doing so. Therefore there must be some value. What kind of value?

The most important one is representativeness: When using a systematic technique, we test representative and special values. We test for two reasons: To find as many problems as possible, and to assess the quality of the product, i.e. we test with destructive values and with representative values.

The other much-needed quality of testing is accountability: We need to be able to tell our clients what we have tested, what not and why. Or what we tested more or less.

When the software works or fails for one value, we assume, very often rightly so, that it will do the same for other values. The basis for this is called **equivalence partitioning**. The benefit: We do not unnecessarily duplicate testing effort.

But how can we do it?

Let me introduce a few methods, to make it easier.

First method: Test one right and one wrong value

In many cases software works relatively well if your inputs are correct or ”mainstream”. If some input is ”special” or wrong, it fails. For consumer software, when confronted with failures, the user will often ask, ”How could anyone have forgotten to test this obvious thing?”

Examples:

- The input should be numeric: Test one typical correct value. For the incorrect one, choose a value which is too high (extremely high for example) or a negative number. Zero is a typical destructive value, often difficult to handle for a program. Trying a non-numeric input is also a valuable choice.
- The input is text: Try a usual text with usual length as correct. Try special characters, a missing text (no input) or a text which is too long as wrong possibilities.
- A group of allowed values: Try the most common one as correct, and something else that is wrong. For the wrong one you could choose a value that is ”near correct” or that was correct in an earlier version or is correct in other software.
- In any case: Try to leave out inputs, and try repetitions.

The value of this technique: You will at least get some impression of how error handling works with your software. Error handling is often badly designed and implemented, and also badly tested by programmers. Thus the wrong values have good value when it comes to discovering trouble.

As you see, this technique leaves many choices, e.g. which of the discrete allowed values you should choose. Or if you should choose a numerical value that is too high or too low. Thus we have another method:

Second method: Equivalence partitioning

Somehow, this is ”the mother of all testing”. The idea is to partition any set of possible values into sets that you think are equivalent. The idea of equivalence means that you think the program will handle equivalent values in, principally, the same way. YOU think! This means someone else may find a different partition. It depends a bit on viewpoint and experience. Then you select one value per equivalence class for testing. IN PRINCIPLE, handling the input in the same way means that you can assume the program is executing the same lines of code. For example, for a bank transfer, it should not matter if the amount is 100, 140, 145, 150, or something like that. Or for checking validity of a date input, at least every day between 2 and 27 should work the same way.

A different use of this technique is backwards: When I review test plans, I look at the values planned to be used in the test. Then I try to figure out what equivalence classes might be behind them, which other equivalence classes there may be, and then I search for the other test values covering these other equivalence classes (and often find ”holes in the test”).

Here are the main rules:

- If an input is numeric, choose a value that is too small (wrong); one that is too high (wrong) and one that is correct (right).
- If an input is logical, then test both true and false.
- If an input is a time value, do as with numeric values, but include a fourth class: impossible values (like 35th of December).
- For discrete inputs, test every allowed value plus one wrong.
- Test every input in two ways: given and not given.
- Try correct and wrong data type for inputs. (Example: texts in a number field, Chinese characters for an ASCII text field etc.)
- If an input can be repeated, try zero, one, more than one repetitions.
- **The pessimist rule**: If you think some-

thing might not be equivalent, then partition any such equivalence class into subclasses. (This creates a more thorough test, but also more testing work). Real pessimists would test every value, but then they would never finish testing.

If you look at other systematic testing techniques, many of them are based on equivalence classes.
This technique is easy to learn and use, and it saves a lot of duplicated test effort. You test one value per equivalence class only. Combinations of equivalence classes may be tested as a more advanced technique. For most inputs, if the data type and the conditions are given, equivalence partitioning is easy: It is the same over and over again. In principle, just use the checklist above!
There are tools, which support the technique. CASEMAKER is one of them.

The trouble with this technique:
- Logical combinations are not very well covered.
- The technique is blind for trouble when the implementation of the program is done in a completely different way than you think it may be.
- The technique may be too complicated when experienced people use the pessimist rule a lot.
- Some values may find more problems or may be more representative than other values.

Therefore we may need more. This is described in the remainder of this paper. There are several techniques.

Third method: Logical combination testing

Logical combinations are dangerous. In many cases there is unclear thinking: Some combinations are forgotten. Distributed logic, implemented in different places, may have holes or side effects. The conclusion is: Logical combinations must be tested. The problem: Combinatorial explosion. The number of tests would grow by the power of two per new combination. This may be OK for small problems, but if the number of parameters exceeds about five, this tends to generate too much work.

As an example, here is the table of all pairwise combinations of three logical inputs:

Test case number	Input 1	Input 2	Input 3	
1	N	N	N	
2	N	N	Y	
3	N	Y	N	
4	N	Y	Y	
5	Y	N	N	
6	Y	N	Y	
7	Y	Y	N	
8	Y	Y	Y	

Every new variable will duplicate this table!

One way to cut down on this is testing only lower-level combinations: Observation has shown that pairwise combinations detect most errors. Less errors by combinations of three values, and so on. There are many tools supporting pairwise combinations, where all pairs of values from different parameters are generated automatically. The technique is popular, as it saves a tremendous amount of work.

As an example, here is the table of all pairwise combinations of three logical inputs:

Test case number	Input 1	Input 2	Input 3	
1	Y	Y	Y	
2	Y	N	N	
3	N	N	Y	
4	N	Y	N	

However, pairwise combination testing may be dangerous, as it may overlook errors. This is especially true if there are implicit dependencies. An example of such dependencies is the validity checking of a date input: The number of days in a month is dependent on the month and even the year. However, if there are no known dependencies, then pairwise testing is a good first choice. It may even be applied if some values are not logical, but discrete.

Example for general combination testing with discrete values:
One input is the month, another one is a color. The month falls into three categories: 31 days, 30 days, and February.
The colors may be red, blue, black, and white. All combinations of these two inputs are 12.

A more difficult technique is cause-effect graphing. This technique is only useful if supported by a tool. Otherwise it is too difficult to apply. The analysis may introduce more problems than are found.

There are more techniques available for logical combination testing, but these would be too complicated for a short article.

Fourth method: State-transition testing

Many programs are defined as state-transition diagrams, state charts, or state machines. The technique is popular and supported by UML. A corresponding testing technique exists. It was defined back in the 1960s and 1970s.

Examples for state machines are:
- A mobile phone. It may be on, off, may be a phone, a camera, a database, and SMS machine etc.
- A traffic light: The state is often visible as the displayed light.
- Most home appliances. Buttons introduce state transitions.
- The status accounting in a defect reporting and tracking system.

These are the different coverage criteria:

1. The test reaches every (reachable) state
2. The test passes through every (reachable) transition
3. The test should cover every possible event in every state
4. The test should cover combinations of transitions after each other.

All these test design methods are easy to apply if the state machine is defined in a formal, machine-readable way, and if a test design tool is applied. Actually, most of the model-based testing is about testing state machines.

State machine testing tests only the state machine logic in itself. If data can vary, they must be tested using other techniques, like equivalence partitioning (boundary value analysis (see later) or logical combination testing).

The first method is not very valuable. The main value is in finding states that are not reachable, even if they should be, or requirement problems (misunderstandings, unclear requirements etc.). From the start state or from initialization, the state machine is fed with inputs in order to get it to assume every state. If some state seems unreachable, the problem is researched.

The second technique has a value: It checks if the state machine "works", i.e. if every defined transition gives a reasonable result. An example is the answer to the question what should happen if someone presses this or that button. The test design starts again from the start state, and now in every state, every defined transition is tested. Typically, these are the "valid" transitions. An example would be to switch on your mobile phone, and then test everything there is in the user manual that has to do with state (e.g. the menu selection).

The third technique will find that not every possible event has a defined reaction for every state. The technique described above, testing one right and one wrong, is an example of this: The "wrong" events are often forgotten. However, if there is a "water-tight" specification, testing in every state has probably been done already. To continue the mobile phone example, this technique would require trying every button in every state of the phone (considering only buttons as inputs here).

The last technique is difficult and possibly leads to indefinite amounts of work: One may test two, three, n events after each other, starting from each and every state. This kind of testing finds "long-term corruption", i.e. the state machine somehow depends on several events after each other, and on the order of these events. Even hidden extra states implemented may be found. But the amount of work for this is probably too much. This technique, if applied at all, requires tools.

However, a "trick" may be applied: When testing state machines, one test should be chained after the other, with no restart in between. This will at least cover some of the combinations

Save the date!

TESTING DAYS GERMANY 2008

Compuware and Diaz & Hilterscheid invite you to attend a free event series for testing professionals in autumn 2008:

- **26th September · Duesseldorf**
- **29th September · Frankfurt**
- **30th September · Munich**
- **1st October · Hamburg**

Presentations include a mini tutorial on test design by „Mr. Test“ Hans Schäfer, case studies and much more. Events will be followed by a get together for testing professionals to exchange experiences.

More details on www.compuware.de/testingdays

Compuware

and thus be more destructive.

In the mobile phone example, we may try to press buttons one after each other, in all possible different orders. (People who tried this sometimes had to take back their phones for service). With these mobile phone examples, I ignore that the phone has some more states, depending on incoming calls etc. Thus, in reality, the test is even more complicated than outlined here.

Fifth method: Boundary value and domain analysis

Some values are better than others in finding trouble. Boundary values have this property. This is because people have problems expressing boundaries correctly, and because programmers tend to use the wrong comparison operator (less or equal instead of less, for example). Another problem is counting: Boundary value analysis is good against off-by-one errors.

Boundary value analysis can very easily be combined with equivalence partitioning. When choosing values in an equivalence class, the boundaries are prioritized.

The technique asks us to test two or three values near the boundary: The boundary itself and one or two values as near as possible either side of the boundary. I used to test all three of them: Below and above the boundary as well as the boundary value itself. This requires less thinking and is safer. I call this ”the Friday afternoon method”. If you have less time and experienced people, you may just test two values: The boundary and another one nearby which shall be in the other equivalence class. This requires more thinking. I call this the ”Monday through Thursday method”.

When it comes to boundaries in several dimensions, like the border of a window, you test each linear boundary with three values: Two (on n, for n-dimensional problems) **on** the boundary, one inside or outside, in the other equivalence class. Domain testing is interesting when boundaries are not geometrically vertical or horizontal. Then you save a lot of combinatorial testing. However, for three or four dimensions, the technique soon becomes difficult.

Boundary value analysis should not only be applied to user inputs. There are many other (more implicit) inputs, and they should be tested. James Bach, a testing guru, defines a few more rules. Boundary value analysis should also be used to

- question the validity and interpretation of requirements,
- discover (implicit) boundaries the programmers overlooked,
- learn about boundaries that emerge from the interactions among sub-systems,
- discover the absence of boundaries where such absence creates the opportunity for performance and reliability problems.

Here is a list of (implicit) boundaries to check up:

Buffer sizes
Table sizes
First and>> last elements in buffers, tables, files etc.
Memory boundaries
Allowed number of repetitions
Message sizes
Lengths of output fields
File lengths
List lengths
Transition in time (over the hour, day, year)
Timeouts

This kind of testing requires programming experience. You need to know what is going on behind the scene.

An example boundary problem in PowerPoint for Mac, WORD ART:

- Fill in characters until the maximum (just hold down some key!).
- When the maximum (about 128) has been reached, no more characters can be input, but no error message is displayed. Keyboard input is just ignored.

What is missing?

There are many more techniques. Some of the most useful or often discussed ones are these:

- Random and statistical testing
- Exhaustive testing
- Error guessing
- White-box testing
- The test oracle

Random and statistical testing

Software is digital. It may fail for any value. Equivalence class partitioning is a black- box technique. It tends to overlook things implemented in the program unknown from the outside. For example “Easter eggs”, special conditions triggering very special behavior. This means, in principle, that any value could be tested and has some chance of discovering trouble. On the other hand, a randomly selected test normally has a low defect-finding ability. It only works if there are very many tests. Statistical testing tries to improve random testing by concentrating on values, which will occur more often in practice. However, in order to apply it, the tester must design a usage profile for the application under test. This may be a huge endeavor or impossible.
Anyway: If anything can easily be tested by generating random inputs, and checking the output can also be automated, then this method is promising.

Exhaustive testing

This is the extreme case: Every possible combination of input values is tested. In principle, this should find all problems. In practice aplying this method is not possible. There are plainly too many possibilities.

Error guessing or fault attack

This method concentrates on typical problems with the program. There is an assumption that the tester has some idea about what typical problems are. Examples include:

- Boundary values (covered by boundary value analysis)
- National and special characters for text values
- Missing and duplicated input
- Too much input
- Low memory
- Unavailable resources
- And more.

A tester should always keep a log about typical problems. Checking the existing defect log is a good idea.

Error guessing is the oldest test method. Over time, much of it has been systematized and has flown into the other methods described here. But a tester should always put in “the little extra” for concentrating on typical problems.

White-box testing

This method is typically used for unit or component testing where there is access to the program code. The tester tries to cover all code. There are many different coverage criteria, but an absolute minimum should be to cover most of the branches in the code. This will assure that most of the statements or lines of code are also executed. The question is: How much do you know about a line of code that is never executed?

The test oracle

Testing requires you to interpret the system’s reaction to the inputs. You compare the output with the expected output. But who tells you the expected output?

The test oracle!

It is defined as a mechanism that decides if a result is right or wrong. In principle, you look into the specification. But often, results are complicated and interpretation is difficult. It would be nice to have an automatic oracle. This would be another program, but it could be a spreadsheet implementation for the mathematics, or an older version of the program. If this is not available, you may test with “easy values”, where manual calculation is easy. Or you might deviate from full checking and just do a plausibility check. In the worst case, the only thing you do is check that the program does not crash.

Summary

Test design techniques are valuable. You have to design your test cases anyway, so why not do it systematically. The benefit is: If somebody asks how you did it, you are able to describe it, plus your reasoning behind it. Your test will be accountable. And you may be able to improve over time.

To randomly fire some shots at the program is not testing!

Biography

54 years old. Specialist in software testing
1979 M. Eng.from Technical University of Braunschweig, Germany. Computer science, railway signalling
1979 - 1981: Development of real time process control software at the Fraunhofer-Institute in Karlsruhe, Germany.
1981 - 1987: Work with Center for Industrial Research in Oslo, Norway. Development of parts of an IDE-tool (Systemator). Later developing test tools and leader of quality improvement program. Consulting and lecturing about software quality.
1987 - 2007: Independent consultant in software testing and testing improvement matters. Guest lectures at several universities in Norway about Quality Assurance and Software Testing. Public and in-house seminars in Software Review and Testing in Scandinavian and European countries. Regularly speaking at conferences. Several best paper awards. (Best presentation award at CONQUEST 2003, best paper in 2004).
1998 - 1999 Test coordinator in Norwegian Telecom's Y2K project.
1999 ISEB Certified Tester (Foundation Level)
2004 ISTQB Certified Tester (Foundation and Advanced Levels)
2003 - 2007 Chairman of Norwegian Testing Board
1992 - 2007 Active participant in running museum trains in Norway, certified for safety critical services on steam locomotives.

Noch Fragen?
Sie erreichen uns unter:
+49 (0) 711/ 972-80000

Egal, wo Sie herkommen: Wichtig ist, wo Sie hinwollen.

Die Deutsche Telekom ist in Bewegung: Gestalten Sie den Wandel der vernetzten Welt mit – in einem der führenden IT- und Telekommunikations-Unternehmen. Bundesweit suchen wir Senior Testmanager/Projektleiter und Senior Consultants, die unsere anspruchsvollen Themen und Projekte mitgestalten. Unser Wandel ist Ihre Chance!

Arbeiten bei der Deutschen Telekom
- T-Systems ist die Geschäftskundensparte der Deutschen Telekom und bietet hochwertige Dienstleistungen und integrierte Lösungen für Informations- und Kommunikationstechnik (ICT) aus einer Hand.

Senior Testmanager/Projektleiter-Test Factory (m/w) ID 2008-3615

Das erwartet Sie bei uns
- Verantwortung für internationale Testprojekte
- Leitung unserer Testteams (onsite/offshore) sowie Steuerung der Testplanung, -koordination und -durchführung
- Fachliche Personalführung der Projektmitarbeiter
- Durchführung des Projektcontrollings
- Unterstützung unseres Fachvertriebes bei Presales-Aktivitäten, wie z.B. fachliche und inhaltliche Angebotserstellung

Das erwarten wir von Ihnen
- Mehrjährige Erfahrung im Management von Testprojekten
- Kenntnis der Testmethodik und Testtools sowie QS-Verfahren
- Kommunikationsfähigkeit, hohe Kundenorientierung
- Gutes Verhandlungsgeschick
- Verhandlungssichere Englischkenntnisse und eine hohe Reisebereitschaft

Senior Consultant Test Factory (m/w)

ID 2008-6750

Das erwartet Sie bei uns
- Testconsulting für internationale Projekte
- Erarbeitung geeigneter Strategien und Konzepte zur Optimierung des bestehenden Testprozesses und Beratung zur Aufbau- und Ablauforganisation von Testprojekten
- Unterstützung unseres Fachvertriebes bei Presales-Aktivitäten

Das erwarten wir von Ihnen
- Branchen-Know-how aus einem unserer Zielmärkte (Public, Telekommunikation, Automotive)
- Erfahrung im Testconsulting oder Testmanagement
- Kenntnis und praktische Anwendung von Standards und Methoden wie z.B. TPI®, CMMI, ISTQB, ISO 9126
- Kommunikationsfähigkeit und Kundenorientierung
- Verhandlungssichere Englischkenntnisse und eine hohe Reisebereitschaft

Wir freuen uns auf Ihre Online-Bewerbung mit Angabe der Stellen-ID unter www.telekom.com/your-chance

Nähere Informationen zur T-Systems Enterprise Services GmbH unter www.t-systems.de

Erleben, was verbindet.

Test Process Improvement using TMMi

by Erik van Veenendaal, Rob Hendriks, Jurian van de Laar and Bart Bouwers

Introduction

More and more organisations make efforts to improve their software development processes. The reference model that is most often used is the Capability Maturity Model Integration (CMMI). In this model practices that are related to verification and validation activities are described, but the level of detail is too limited from the viewpoint of the test professional. To fill this gap, the Test Maturity Model Integration (TMMi) has been developed by the TMMi Foundation [www.tmmifoundation.org], using the TMM framework as developed by the Illinois Institute of Technology as one of its major sources. The TMMi provides a structured approach for test process improvement. Testing as defined in the TMMi is applied in its broadest sense to encompass all software quality-related activities. Within Europe the number of companies using the TMMi is increasing.

Practical experiences are positive and show that the TMMi and its predecessor TMM support the process of establishing a more effective and efficient test process. Testing becomes a profession and a fully integrated part of the software development life cycle. Applying the TMMi maturity criteria will improve the testing process and has a positive impact on product quality, test engineering productivity, and test execution lead-time.

In this paper we give our recommendations on how to organize and execute a successful test improvement project, using the TMMi as a reference. We've gathered the practical do's, don'ts and examples that we consider to be the most important success factors, based on our own experience. For a complete description of the TMMi and its guidelines we refer to the web pages of the TMMi Foundation.

Overview of the TMMi maturity levels

Just like the CMMI staged representation, the TMMi has a staged architecture for process improvement. It contains levels that an organisation passes through as its testing process evolves from one with an ad-hoc and unmanaged nature to a mature and controlled process with defect prevention as its main objective. Achieving each level ensures that adequate improvements have been made as a foundation for the next stage. The internal structure of the TMMi contains testing practices that can be learned and applied systematically to support quality improvement in incremental steps. There are five levels in the TMMi that define a maturity hierarchy and an evolutionary path to test process improvement.

Each maturity level contains a comprehensible set of process areas. The process areas for maturity level 2 are shown in figure 1. Experience has shown that organisations are most successful when they focus their test process improvement efforts on a manageable number of process areas at a time. Because each maturity level forms a necessary foundation for the next level, the decision to skip a maturity level is usually counterproductive. On the other hand, test process improvement efforts should focus on the needs of the organisation in the context of its business environment.

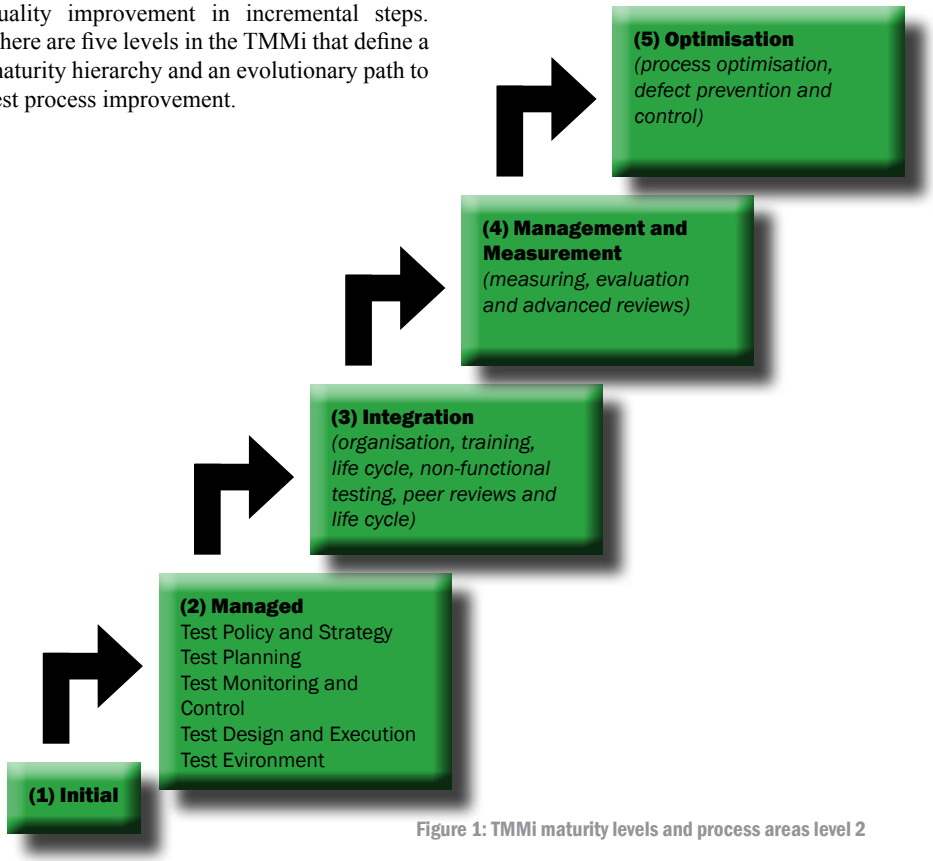


Figure 1: TMMi maturity levels and process areas level 2

It is possible that a process area at a higher maturity level may address the current needs of an organisation or project.

The TMMi improvement project

The TMMi is not just a theoretical model but a well-defined set of process areas and goals based on practical case studies. Although the model is quite easy to understand, implementing it in an organisation and thereby improving the test process is not always a straightforward task. On average it takes approximately two years to achieve TMM(i) level 2. To support organisations doing test process improvement, a number of recommendations are provided, based on our practical experiences. They have been major contributors towards success in a test process improvement project. A typical TMMi improvement project has the following phases:

- Initiation (determine goal and scope)
- Planning (define the project in terms of structure, resources, ownership)
- Implementation (developing the deliverables, like procedures and templates)
- Deployment (publication and roll-out of deliverables, institutionalisation)

INITIATION PHASE

Determine the current status

"If you don't know where you are, a map won't help." (Watts Humphrey). Before starting an improvement project it is necessary to know which maturity level the existing test processes have. Therefore the current situation has to be assessed. A formal assessment may not be required to determine the current status; a quick-scan is usually sufficient. A small group of people in the organisation is interviewed and the current set of documentation is collected and checked. The result is a report containing a current *maturity profile* (see figure 2 for an example) and *recommendations* for improvement. The report is presented to management and this is usually the basis for a TMMi improvement project.

Process Area	Score	Conclusion
Test policy & strategy	3.9	Not satisfied
Test planning	4.6	Not satisfied
Test monitoring	4.2	Not satisfied
Test design & execution	5.4	Partly satisfied
Test environment	6.6	Partly satisfied

Figure 2: TMMi Maturity Profile

Determine the goal

It is desirable to clearly define the goals of the improvement project. These goals should be directly related to the organisational business goals to enable management commitment. If you want to get people involved and committed, you will have to explain to them where the improvement efforts should lead. Adherence to the TMMi is not a goal in itself. The model can provide guidance and practices in achieving a higher maturity in testing. In the end, a mature testing process should contribute to the business goals of the organisation. Predictability, higher productivity, efficiency and effectiveness are examples of goals that an organisation may want to achieve with test improvement. The organisation itself will have to determine exactly which goals to strive for. For one of our customers we have developed a 'vision document' in which we described the envisioned future as an end goal of the improvement initiative. Creating such a document is important to reach a common understanding about the 'why' of the improvement project, to get commitment and to guide the improvement project along its way.

PLANNING PHASE

Organising the project team

A common problem is that improvement initiatives get insufficient attention and resources to really make a difference. We have seen very positive results when the project is organized around the quick scan's recommendations. Each recommendation is assigned to a *Recommendation Workgroup*. The workgroup implements the improvements and deploys them. Since the changes are to be deployed in test projects, each workgroup consists of representatives from all test projects. Each workgroup reports to the test improvement project manager, who in turn reports to the steering group. It is crucial that the steering group has members that will support the improvements and benefit from them. If no sufficient capacity is allocated to staff the improvement project team, and the team members are not enabled to spend time on improvement activities, then all resources will soon be occupied with their other daily operations instead of improving the process.

Communication

One of the most important parts of the improvement project is communication. Improving means introducing change. People will only change if they really understand why change is necessary, so inform them, continuously. People see their colleagues participating in the improvement project and want to know what is going on. They, as well as management, must be informed on a regular basis. Publishing periodic newsletters showing the progress and having regular department meetings are ways to do this. Another best practice is to involve these persons in reviews of the project's deliverables. In other words: share what you are doing to create a buy-in. You can document your approach in a communication plan, e.g. with a mind map.

Process ownership

With their knowledge and experience, external consultants can make a valuable contribution to an improvement project. Especially if they are allocated to the improvement project on a fulltime basis, they can really be the stimulators of process improvement in terms of effort and progress. In one project we experienced the difficulties of combining the responsibility for an improvement project with the role of test manager, especially when operational issues constantly demand attention. We also experienced the benefits of working in pairs: the external consultant teaming with the internal test manager. We discovered that with a good balance between process and project, the improvements were immediately embedded in the organisation. Inevitably the external consultant leaves, and then the organisation must be able to continue the improvements on its own. Therefore we recommend that the ownership of the improvement project and the test process is given to the internal staff, not to the consultant. A common pitfall of consultants 'driving' the improvement project is that they fail to adapt to the maturity level of their customer. Some organisations want to use their own strengths to achieve improvements: to what extent do they accept the involvement of an external consultant? It is difficult to achieve real commitment when the changes are (perceived to be) 'forced' from the outside. Make clear agreements on responsibilities, also for a consultant it is important to know what he is accountable for and what the scope of his involvement is.

Create stakeholders involvement

Not all stakeholders may be obvious at first sight. Therefore we recommend a stakeholder analysis at the beginning of the improvement project. The people whose participation or contributions we need are stakeholders, but also the people who should have a certain interest in our results. Management, or more specifically the sponsor or champion who supports the improvement project, evidently needs to be informed and involved. Also think about quality assurance, the owner of the quality system and those responsible for the test infrastructure. Other disciplines may be involved or affected as well, such as a tailored test training for developers.

The maturity of the development processes

It is important to not only look at test processes but also at the processes which testing is highly dependent on. Without mature supporting processes it will be very difficult or even impossible to improve the test process. Make sure that project management, configuration management, requirements management and risk management get enough attention in the organisation that you are improving. Lack of project management will cause test management to fail. If the organisation is not used to working on projects, it will be difficult to embed the testing activities in a structured way. When applying test design techniques it is recommended to have a set of requirements. Also changes in the requirements should be well managed and the test team should be notified. Finally, not having a configuration management system often results in defects that cannot be reproduced and defects reoccurring that were supposed to be solved. These types of problems can't be solved within the test process itself. Consider starting a separate improvement project for these supporting processes, or just don't start a test process im-

Register Now!

www.conquest-conference.org

25 % discount for testing experience readers

The conference program is included in this issue of testing experience.

- Service Oriented Architecture
- Business Process Engineering
- Secure and Safe Software-Based Systems
- Secure Software Development
- Model-Driven Engineering
- Requirements Engineering
- Verification and Validation
- Testing
- Metrics and Measurements of System Quality and of Development Processes, Analytical Models of Software Engineering
- Project Management
- Configuration Management

Keynotes

- **Libby Affen**, *Matrix Global Israel (Talpiot), Israel*
- **Bjoern Brauel**, *Software AG, Germany*
- **Andreas Kindt**, *Deutsche Telekom AG, Germany*
- **Ingolf Krüger**, *University of California, San Diego, USA*
- **Andreas Spillner**, *Hochschule Bremen, Germany*

& **Karin Vosseberg**, *pdv.com Beratungs-GmbH, Germany*

Partner Country

ISRAEL

Organizer

iSQI
International Software Quality Institute

Your Contact

Tanja Brütting
tanja.bruetting@isqi.org
Tel +49 9131 91910-16

info@isqi.org
www.isqi.org

provement project at all if the organisation is not ready for it.

Alignment with other improvement initiatives
Within organisations other improvement initiatives can be on-going, e.g. software process improvement based on the CMMI. Beware of the 'yet another initiative' syndrome! A department can only handle a certain amount of change at a time. This highly determines the change velocity. To prevent that duplicate activities are executed it is recommended to map how the test improvement activities contribute to other improvement activities and vice versa. This mapping should aid project managers in working with improved processes in their projects.

IMPLEMENTATION PHASE

During this phase all planned changes, typically based on the recommendations from a quick scan, are implemented. When possible, use existing processes and best practices as much as possible. Sometimes a thorough process is already common practice in the organisation; it just needs to be documented.
Although in practice probably all process areas of the applicable maturity level will have to be addressed, this paper will only focus on the process areas that are essential for making a good start. At maturity level 2 the test strategy and policy are the first and most important subjects that give direction to the entire improvement project. This is one of the strengths of the TMMi: its top-down approach that starts with the organisation's view on testing, and using this as the basis for further improvements. Metrics are important to measure the effect of the changes and drive further improvements.

Test policy
Within the TMMi the process area 'Test policy and strategy' requires that an organisation defines a test policy that states the organisation's view on testing. It answers questions like 'Why do we test?'; 'How important is quality?'; 'Are we time-driven or quality-driven?'. Typically the answers to these questions are related to the business goals or strategy of the organisation. The test policy is an important basis for (management) commitment to all the activities in the organisation related to quality and testing. In one of our improvement projects, for example, we derived the test policy directly from the so-called "One Page Development Strategy" that was already available and deployed at a higher organisational level. However, keep in mind that a test policy should be clear and concise. The business language that is typically used in mission statements and business strategies is often too abstract to clearly explain what testing is about. It is better to use testing terminology, e.g. by summarizing the main themes that have focus in the organisation, like 'risk based testing', 'test training' or specific (non-)functional quality characteristics, like reliability and usability.

Test strategy
A test policy is implemented in the organisation by a test strategy. The TMMi requires the definition of a test strategy for each programme in the organisation. This strategy provides clear insight into how the products are tested throughout their life cycle. It has to fit into the organisation's development process (e.g. RUP, RAD, V-model, DSDM). The strategy contains a definition of test levels and the organisational groups that execute them, the approach to regression testing, entry and exit criteria, and more. A sample strategy is shown in figure 3. Make sure that you create a strategy that is easy to explain to stakeholders, mainly your testers. The strategy must be aligned with the organisation's test policy.

	Unit testing	Integration testing	System testing	User acceptance testing
Objective	Unit meets its design. Ensure low level quality. Unit tests run automated for each build (regression test)	System is ready for system test: quick functional check. Integrated units meet interface specifications/global design/architecture.	System meets specification (FD). System does not show regression.	System meets specification (FD). System meets business requirements and expectations. System is "fit for use" in its business context. System does not show regression.
Responsible	Development	Development	System test team	Acceptance testers in business units
Entry criteria	Developers are challenged to write unit tests by TLs and Seniors, following a unit test strategy.	All unit tests pass.	FDs approved. All unit tests pass. No open defects with severity major or higher.	Requirements system tested. No open defects with severity major or higher.
Exit criteria	All unit tests pass (build does not fail).	All unit tests pass. Planned code reviews executed and rework done. No open defects with severity major or higher.	100% Test cases executed. No open defects with severity major or higher.	System meets acceptance criteria. No regression. Business approval for production deployment.
Techniques	Ad hoc testing.	Ad hoc testing / error guessing	Black box techniques. Error guessing.	Error guessing. Some black box techniques.
Tools	JUnit	Jira	HP Quality Center (Test Director + QTP) Jira	Jira

Figure 3: Sample test strategy for a V-model process.

Metrics
Measuring the progress and quality of testing as well as establishing performance indicators for the test process are important to monitor the status and to show the benefits of the improvement process. Think about metrics in the area of efficiency, effectiveness and deployment. A common pitfall is that a (too) large number of metrics is defined. As a result the collection of measurement data requires a lot of effort and we often see that the collected data is not used or analysed. Goal-Question-Metric (GQM) is a good method to define a useful set of measurements. With this method every metric is derived from a specific goal and implements a specific question from a stakeholder related to that goal. Thus a metric is only defined if there is a clearly related need for it. Keep metrics as simple as possible!

Documentation structure
In an early phase of the improvement project a document overview should be established. In one TMMi project the documents were initially delivered separately by different teams for their particular improvement action. When all individual documents came together it was obvious that an overall document, tying all parts together into one total procedure from the start to the end of the test process, was missing. Once this overall procedure had been developed, it became the starting point of a structure or hierarchy for all the other documents in

the quality system. By thinking about a solid structure for your documents at the beginning of your project, you can save a lot of time on reworking and restructuring later.

DEPLOYMENT PHASE

Accessibility of deliverables

Make sure that all documents, both procedures and templates, are easily accessible for all testers and other stakeholders. An intranet page is a very effective way to provide a clear, preferably graphical structure, allowing users to easily access and download the documents they need. In one project we depicted the documentation structure as a graphical presentation of a 'pyramid', with the overall process at the top, and all related and supporting documents, such as work instructions and templates in the basis. This test framework became a familiar artefact in the quality system and soon everybody recognized and used 'The Pyramid' on the intranet as the starting point in their search for a document or template.

The change process

Deployment is the most difficult and time consuming part of the improvement project. You can publish a procedure or template, but that doesn't mean that it will be immediately used by everyone. Often the problem in improvement projects is not the availability of documentation but the lack of adherence to the described way of working. To achieve a maturity level, at least 80% of the people in the organisation should work according the documented process. If existing procedures are not followed, try to solve the root cause first. Only when it's clear why current processes are not being followed is it worthwhile to change and enforce them.

The throughput time of the change process does not only depend on the amount of changes, but also on the organisation, its culture and the running development project(s). It may be wise to look for a suitable pilot project to see if the changes have the intended effects and if the procedures are really feasible in day-to-day practice. But such a pilot candidate must be available – preferably in one of its early stages – and willing to participate. Also in an ad-hoc culture, where local heroes have a large influence, changes leading to a more structured and disciplined way of working may not be directly embraced. When applying changes, be prepared for resistance. Don't change more than necessary, take small steps and communicate the changes as often as needed and in any way you can think of. An improvement project is not successful until it has been deployed by people motivated to improve.

We thank Richard Grooff for his contributions to the initial versions of this paper.




Biography

Erik van Veenendaal is the founder and director of Improve Quality Services BV. He is an internationally recognized testing expert with over 20 years of practical experience. He is the author of numerous papers and a number of books, including the best-sellers 'The Testing Practitioner', 'ISTQB Foundation' and 'Testing according to TMap'. Currently Erik is the vice-president of the ISTQB and the vice-chair for the TMMi Foundation. For his contribution to the testing profession he received the "European Testing Excellence Award" in December 2007.

Rob Hendriks has more than 12 years of experience in software quality, with great emphasis on software testing. The last few years he has been active as a senior consultant in projects for consumer electronics, professional technical systems and administrative systems. He regularly runs courses in the area of inspections and testing (ISEB Foundation and ISEB Practitioner accredited), and is specialised in the field of test design techniques. Currently he fulfils the role of operational manager within Improve Quality Services Ltd.

Jurian van de Laar has more than 13 years of experience in software engineering. As a senior consultant at Improve Quality Services, he is involved in several TMM/TMMI and CMMI improvement programs. He had a leading role in achieving TMM Level 2 at Philips Healthcare in 2007. Jurian is an accredited teacher of the ISTQB Foundation and ISEB Practitioner training. He is member of the Syllabus working party of the Belgium and Netherlands Testing Qualifications board.

Bart Bouwers is a Senior Test Consultant at Improve Quality Services. He has over 13 years of experience in software development and software testing, in several positions, in both technical automation and information systems domains. Bart is a certified ISEB Practitioner. Currently he is running a TMM Level 2 improvement project at a Dutch bank.

Interview Norbert Bochynek

Director IT Strategy

Not every reader of this article is familiar with DSGV. Can you please give us a brief overview of the tasks and goals of DSGV?

The Deutscher Sparkassen- und Giroverband (German Savings Bank Association, DSGV) is the umbrella organization of the Sparkassen-Finanzgruppe. This includes 446 savings banks, 7 Landesbanken, 10 Landesbausparkassen, 12 public insurance companies and many more financial service providers.
DSGV represents the interests of the Sparkasse financial group, organizes the formation of intention within the group and determines the strategic direction. For this purpose it partly operates in an affiliation with the regional associations and additional institutions of the Sparkasse financial group.
DGSV is the carrier of the central educational facilities of the Sparkasse financial group: the German Sparkasse Academy and the college of the Sparkasse financial group, the University of Applied Sciences. Other community institutions are the Association for the Economic Support by the Sparkasse financial group, the Eberle-Butschkau Foundation and the Sparkasse Foundation for International Cooperation.
It also administers the institute-securing programs according to the Deposit Protection and Investor Compensation Law and the liability association formed for this purpose, along with the insurance funds of the bank transfer centers and the insurance funds of the Landesbausparkasse banks.

As the director, you are responsible for the IT strategy of DSGV? In what way are you supporting your member institutions?

My department provides support during the rendering of nationwide IT concepts. These serve as recommendations for the member institutions but are not legally binding. Among other things, my division implements projects to develop control systems for IT, for the invoicing service of Sparkasse and for the trade with the used licenses.
We also coordinate the implementation of regulatory requirements (e.g. the compensation tax) as well as the business strategies (financial concepts) that are developed at DSGV to IT. Here we are working especially closely with FinanzInformatik.
Employees of the institutions, FinanzInformatik, and association partners are collaborating on all of our projects and look for the best solutions and concepts.

Which challenges is your department facing at the moment?

The fusion of FinanzIT with the Sparkassen Informatik processing to form FinanzInformatik has been completed. The consolidation process of ten data centers in 1998 to one in 2008 has therefore been achieved successfully. This fusion makes it possible for us to achieve sustainable synergies and these now have to be increased.

The development in the IT of the regional banks will be changed significantly by the current influences. Until there is clarity here about the business models and fusions, not much will change in the IT collaboration between the regional banks.

Which challenges do you see particularly for the IT in this crisis among the (affected) banks?

The cost and margin pressure as well as new legal requirements by the legislator will have the effect that banks will have to pay closer attention to the costs, optimize their IT and handle their project challenges much more effectively. The optimization will not only lead to the implementation of new and improved systems but also to the slimming down of the processes and a corresponding training of the employees. One bank or another will surely look for near-shoring and off-shoring possibilities and accordingly keep the costs within certain limits.

> **The development in the IT of the regional banks will be changed significantly by the current influences.**

IT is decisive for the success, regardless of whether a bank is more or less affected by the subprime crisis. We can see that some banks have a clear economic advantage compared to others due to their superiority in IT. A good example in Europe is the Spanish Santander Group.

You have addressed a few subjects that I would like to examine individually.
You're speaking about the use of new, improved systems. We're observing that some of the new and prevailing systems and applications in the banking world still don't have the necessary maturity, which has the result that the old systems have to continue to be operated simultaneously at least for a while. Doesn't this result in higher costs after all and make IT inefficient?

Well, there will always be problems during the employment of new technologies and systems. There is no IT project that is not faced with these challenges. With some projects the problems can be solved in the prescribed or planned time period, but not with most of them. The planning of IT projects has to be improved. Everyone in IT has to face this

problem – it's not a specific problem for a particular group.

About the other point: it is known that there are providers who are not offering "the heart of the matter" and that their systems don't perform in the way they are supposed to in the end. But I think that the market will be regulated on its own even in this respect. No IT management board will want to carry additional costs in the long-term. The effectiveness and efficiency of the IT added value chain is a central goal. One might believe that some providers and IT executives think that the delivery date is more important than the quality. But in actuality the total costs of systems that were not tested sufficiently are much higher than those of systems that were subjected to a professional and sustainable quality assurance. These additional costs then strain the budgets in the IT business and of course the yields of a bank.

The effectiveness and efficiency of the IT added value chain is a central goal.

There are certainly also deadlines that definitely have to be met. But here the cost/use ratio should be seen in relation to the risk that is being created. In the past we have seen that decisions were made based on a missing risk analysis that afterwards had very costly effects. But I think that the IT managers and stakeholders are aware of this situation.

You spoke about the processes. The effectiveness of IT should be supported by new, adjusted processes, among other things. Do you rely on international standards or models for this purpose in your association?

We support our members with the studies and results from pilot projects and benchmarking analyses of other banks. We don't close ourselves off towards trends or new discoveries and developments.
CMMI, SPICE and especially ITIL are in a boom, sometimes with more or less intensity. I think that we are well-advised to provide our members with the necessary support in IT process management. In this area we work together with very experienced partners so that our member institutes remain competitive.
But the processes are just one side of the coin. The employees and their know-how in filling these processes with life are the other, almost more important aspect. They have to be trained, supported and coached professionally so that these changes have a sustainable effect on the processes. In addition, an effective IT process management is indispensable that effectively controls the procedures with the aid of suitable measured quantities – so-called key performance indicators (KPIs) - and therefore continuously improves them. This is an incredibly important aspect in the quality management of banks.

The lack of qualified IT personnel will increase over the next years.

Quality management and quality assurance play a central role especially in large migration projects. To what degree can you prescribe to the member institutes how the quality management has to be arranged or that they have to use established methods to determine test cases?

We can't prescribe anything and that is intentional. But as an association, we can define certain approaches that concern the entire association. If you find a majority here and they are finalized, then they're usually implemented too.
The persons in charge of IT at the Sparkasse financial group are aware of how much significance the quality of systems and applications has.

We recommend to our members that they support their employees during their daily work, e.g. through targeted continuing education. Two perspectives are very important here: For one, the employee expands his personal profile and is developing it in a future-proof manner. For another, the loyalty to the company as well as its character are supported. The lack of qualified IT personnel will increase over the next years. If one's own employees are not offered the opportunities for continuing education or if they are not trained appropriately for their tasks, there is the danger that they will leave the company to develop their future elsewhere.

You mentioned the training and education of employees to guarantee a sustainability of the changes in IT, to have the employees properly complete their tasks and to form a good connection to the company. There are movements on the subject of certification on a worldwide level; the offers are very diverse. What is your opinion about this?

As I said, it's important to me that the employees are qualified for their tasks. This is the prerequisite to counter certain risks in the life of the project.
All regular testers and test managers should at least receive a basic training – and by this I don't just mean the IT staff but also the specialized departments. The ISTQB certified tester training with almost 100,000 certified people worldwide forms a foundation for this. At the same time everyone is also aware that a certificate alone isn't everything. The em-

ployees have to practice and collect their experiences and here professional coaching is very helpful. What is important in this respect is that the colleagues have initially been told what they can do and should do and how it is done – professionally. The continued development of each individual – like other continuing education – is planned by the respective supervisors. Several steps are provided in the ISTQB certification series that round off the performance of a professional tester and test manager.

it's important to me that the employees are qualified for their tasks

Other relevant certifications are offered as well, such as e.g. according to IREB for professional requirements engineering or the certificate developed by ISSECO on the topic of application security. The certification of ISSECO is something that IT has needed for a long time already. Aside from that ISSECO was founded by German companies, among others, at the top of which is SAP AG. The transfer of these special skills has to be regulated and it especially has to be kept at a high standard.

I would like to emphasize again that the certification alone isn't everything. A certain practical experience is required. With QAMP the certifications are supplemented and rounded off by this important but previously missing perspective. Because QAMP certifies that the person has not only attended a variety of classes and passed tests but also has at least two years of practical experience in this environment.

What has been done in this area in the last few years was and continues to be necessary.

I would like to bring up application security again. Here most people think of firewalls, penetration, etc., or hardware security. Few see the application itself as a weak point. Do you think that this topic will be booming soon?

Yes, definitely. The attention of IT was always directed towards attacks from the outside – knowing quite well that the enemy can also be on the inside! Especially the security of data at companies and the protection of applications will continuously gain in significance and there are enough examples that show that this is necessary.
This perspective of security had not been widely distributed so far, where one had to defend against attacks not only on the outside but the inside as well. Yes, this is a topic.

For years there has been an increased amount of off-shoring and near-shoring in the private bank economy. Do you also see this as a possible model for the public banks?

Of course. Steps in this direction have already been taken by one or another of the public banks. This is another development that one can't close off to.
Nonetheless this also has to be quite well considered. Depending on which country and cultural circle they enter, they will be confronted with different problems. These can also be mastered, that's certain.
Some off-shoring or near-shoring projects fail for the same reasons as one's own projects locally. Those don't always function either, unfortunately.

The characteristic risks that have their origin in the distance, communication and culture have to be minimized. This specific risk management can only be realized with a partner who is working with the local shoring partner. These partners have to be selected very carefully, but in this respect these projects require a very specific planning.

It also has to be made clear, however, that not all subjects are suitable for an off- or near-shoring project – pertinent experiences have already been made in this area.

Thank you very much for your time and the interview.

You are welcome.

Software Testing – the future?

by Geoff Thompson

When I was set the challenge of writing an article on the future of software testing, having got over the initial creative panic, I was wondering how to start. Then I found the following headlines in two well known online computer journals:-

'BA: software failures key in T5 fiasco'
Computing 8 May 2008
'Update: lack of software testing to blame for Terminal 5 fiasco, BA executive tells MPs'
Computer Weekly 9 May 2008

In summary both articles made the same point. During questioning by MP's Willie Walsh (BA Chief Executive) reported that construction work had not completed in time squeezing the window for any real testing. The management met, identified and agreed the risks and issues this creates and decided that the risk was acceptable to enable a go live as planned.

In hindsight Walsh said "If I were to pick one issue I would have done differently, it is that, having recognised the importance of testing and having designed six months of testing, we subsequently compromised on that."
Computer Weekly 9 May 2008

Why, did it interest me? Quite simply the management of the project did exactly the right thing; project management and risk management were used to assess whether slippage in earlier parts of the lifecycle meant that reducing test activity and timescales would have an impact on the quality of the finished article. Obviously the risk was perceived to be low as a decision was taken to reduce testing and go live on time (a decision now regretted by Walsh).

What I also found interesting was that one of the papers saw software testing as the issue. Initially I thought here we go again, testing put the bugs in etc. but then I understood. I wonder if it was more that there was a lack of the right information from the test organisation that would have enabled the right decision to be made by the BA and BAA management. If the test organisation had a grip on the real issues that mattered to the management team and were able to communicate the real risks maybe a different decision would have been taken, or at least the decision that was taken would have been made based upon the right information, with complete awareness of the impact.

To me, testing's role is the provision of data to enable go live decisions to be made (I have heard this termed as Trusted Advisor and Project Intelligence). Yes we might be involved in the decision but ultimately the decision belongs to the Sponsor and his management team, but the data used in the decision will predominately be derived from test activity and must be sufficient to enable the right decisions to be made.

Given this is a major element of our role; I suggest that we need to get this right or the future of software testing will be very dim. We are in the best position, or at least we should be, in a project to gather real data, and to establish what the right data may be, to enable the accurate reporting of project progress information. Far too often today a test team arrives and sits in its cocoon only appearing at the end of the test cycle to announce they haven't finished but can't give anything helpful to say other than perhaps statements like:

"You can't go live because..."

"We have 150 tests still to run"
"We are still finding errors"
"I wouldn't go live yet"– yes I heard that said by one Test Manager one day – and this was all he said

None of these provide even the slightest hint of what the impact of going live is! How critical are the 150 tests left outstanding? What's the error finding profile? Why is testing still finding errors now? These are just some of the questions that these statements generate but can't answer.

So, if we get the information wrong, wrong decisions will be made. There was however one very good example of a situation where the right information was provided in the wrong way, and therefore ignored. The 1986 Challenger disaster was predicted – if you knew that the Space Shuttle would definitely explode killing all of the astronauts, would you have given the go ahead to launch. NASA did. Several engineers had voiced concerns about the effect of the temperature on the resilience of the rubber O-rings that sealed the joints of the Solid Rocket Boosters (the tubes attached to the side of the Shuttle – that are full of fuel at take off). They argued that if the O-rings were colder than 53 °F (12 °C), there was no guarantee they would seal properly. This was an important consideration, since the O-rings had been designated as a "Criticality 1" component—meaning that there was no backup for them and their failure would destroy Challenger and its crew. They also argued that the low overnight temperatures would almost certainly result in Solid Rocket Boosters temperatures below their redline of 40 °F (4 °C). However, they were overruled, the launch happened, the O-rings failed leaking rocket fuel, and the rest is history.

Interestingly the right information was identified but if I am very critical, using words

like 'no guarantee' instead of 'an absolute guarantee' when the management board were under significant pressure from the US Presidency to launch, didn't really emphasize the facts strongly enough. In the same situation, with the same information, I might well have also decided, like NASA did, to launch. Now I know it's a word thing, but in reality if we can't express ourselves and ensure that our message is understood all the great work we have done to improve the processes we use and align vocabulary will be wasted and we will never grow as an industry and will always be seen as the also ran profession! It's all about communication.

Sometimes, alright I concede, most of the time the information we need to provide is not what the Programme wants to hear, but it's our job to ensure the right information gets through to the right person and that they are able to understand and make informed decisions. We need to be prepared to be the bearers of good and bad news.

If testing can't provide the right information it actually adds little value, and that I suggest maybe the situation we find ourselves in a lot of organisations today, we all work very hard at our profession but no one outside of it seems to understand what we do and why.

Whilst working on one client site I remember interviewing a Test Manager who was most upset that the Programme team had asked him to stop issuing his daily report, for what he described as stupid reasons. When I met and interviewed members of his programme team they said that the report was stopped because from day one the Test Manager had had the project flagged as red, what was requested was that the report included relevant data (such as where the issues lay, what element of the software was not working etc.) or was stopped. The Test Manager saw no reason to include the extra data so stopped issuing it. Even today the Test Manager cannot understand what he did wrong and blames the programme team for the delays in delivery because they did not listen to him!

If Software Testing is ever to be seen as a profession and be respected in the future then we should focus on the information we need to enable the Programme to make the right decisions.

Now I am not suggesting that I know the answer. Each person will have differing requirements, and most importantly their requirements may be very different to what they request individually when asked as part of a team (I call this the pack syndrome – when a group influence individual perspectives and results. If you get separate agreement to an action from each individual in a group of people, when you put them together the actions may and often do change), but it is our job to identify what they are and focus our approach to testing in order to deliver these. We need to be conscious of the pack syndrome.

The data requirement is never static, so we also need to be conscious of the changing needs for information as a project progresses.

To enable the future of software testing we have to influence people's approaches to Test Management and the provision of information? Is it a qualification thing, or maybe standards, there are test models out there and what role do tools play in this arena? The following sections look at each of these individually and their relevancy to the issue.

Software Testing Qualifications

You may or may not know that I have been involved in Software Testing Qualifications for a few years now! Firstly working with ISEB (Information Systems Examinations Board) and lately internationally with the ISTQB (International Software Testing Examinations Board). So I guess I am a little bias, but I believe there is value in them. However, there does seem to be some polarisation in the industry as to the value of these types of qualification, some calling for them to be banned, others suggesting that having got them you are able to sell yourself as a competent tester.

The reality is that a good tester or test manager needs many components of skills and knowledge combined to make them successful. In my view the qualifications account for about 10% to 15% of the knowledge required for the practical application in the workplace.

At ISTQB Foundation level there is a short training course (3 days) followed by an exam. However the quick learners amongst us could very quickly understand what is required and pass these exams, maybe even without ever testing anything, there is no minimum experience requirement to take the Foundation.

Intriguingly ISEB have recently revised their single Practitioner Exam, into three, the Intermediate (a mandatory common module which contains their view of the generic detail that is required for the advanced level),Test Management and Test Analyst, but only prescribe a 3 day training course followed by an exam. There is a minimum entry requirement, but they do not include practical test experience, more a passing of time between taking the Foundation exam.

ISTQB, on the other hand, have just launched their Advanced level role-based qualifications, and like ISEB there are three of them, Test Manager, Test Analysts and Technical Test Analysts. However unlike ISEB there is no common module, as these are role based(for example the role of a Test Manager in a review is very different to that of a test analyst) with a minimum training period of 5 days each followed by an exam, with basically the same entry criteria as ISEB.

So I believe qualifications have a very important part to play in the education and verification of software testers and test managers. But the reality is that the exam candidates leave with an understanding of the basic vocabulary and approach to testing, but in most cases no real practical capability, and certainly no understanding of how to communicate outside of the test team.

Standards

There are a few out there, for example BS7925 – 1&2 (the British standard for Component Testing and its associated glossary), IEEE829 (the Test Documentation standard) etc. etc.

In their favour Standards are developed by a very wide working group (these can be as many as 1000 contributors) who build and review the content. In most cases involvement comes from right around the globe. However, by necessity they are generic and some concessions have to be made to gain global agreement (in fact I understand that the latest version of IEEE829 couldn't be released as planned as the detail was not agreed) and provide one solution for everyone, so they are useful but they can't provide the whole answer.

Standards don't, in any example I have seen, indicate how to provide and what information is needed to communicate effectively with the programme, although they can help. For example in IEEE829, when it discusses entry and exit criteria and closure reports looking at planned activity against actual activity and lessons learned, although it mentions these key elements of a test project it doesn't provide any help in pinpointing the right detail.

They are however excellent process models, that provide guidance on how to establish your test project.

Test Models

There are many models available for testers to review against. Most consultancies will have a Test Maturity Model of their own, most sadly leading to creating a sales opportunity, and not actually there to help the organisation that has been reviewed against the model to improve what they do. The nearest commercial model that really does set out to help companies is the TPI® model provided by Sogeti. However this is now a little out of date; it is based upon their in-house test method T-Map which has been updated and those updates have not been reflected within TPI®.

Probably the best model available today to help ensure good processes in use across test projects is the new TMMi Model. This model has been developed by the TMMi Foundation (www.tmmifoundation.org), an international organisation whose aim is to establish a non commercial test process assessment model for the industry. The TMMi model takes its input from many places including the original Bernstein TMM, and others including the Gelperin and Hetzel's Evolution of Testing Model [Gelperin and Hetzel], which describes the evolution of the testing process over a 40-year period, Beizer's testing model, which describes the evolution of the individual tester's

iSQI®
International Software Quality Institute

ARE YOU ALREADY CERTIFIED?

The International Software Quality Institute (iSQI GmbH), headquartered in Erlangen and Potsdam (Germany), develops internationally accepted certification standards for advanced vocational training in the area of software quality. In order to optimize and safeguard the skills and abilities of software professionals, iSQI was founded in 2004 as a subsidiary of the German non-profit organisation ASQF e.V. and works together with international organizations, e.g. ISO (International Organization for Standardization) or EQN (European Quality Network).

iSQI certifies IT personnel in more than 35 countries. With more than 3000 exams per year in Germany alone, iSQI is one of the most important personnel certifier in the area of software quality. The advantages of personnel certification are primarily the secure, comparable qualification of professionals across national boundaries and language barriers.

The iSQI certification program includes standards such as

ISTQB® Certified Tester, Foundation Level
ISTQB® Certified Tester, Advanced Level (Test Manager, Functional Tester, Technical Tester)
iSQI® Certified Professional for Project Management
iSAQB® Certified Professional for Software Architecture
IREB® Certified Professional for Requirements Engineering
iNTACS™ certified ISO/IEC 15504 Provisional Assessor
iNTACS™ certified ISO/IEC 15504 Competent Assessor
TTCN-3® Certificate
Certified Innovation Manager
iNTCCM® Certified Professional for Configuration Management
QAMP® Quality Assurance Management Professional
iNTCSM® – International Certified Professional for IT-Security Management
ISSECO® – International Certified Professional for Secure Software Engineering
T.I.S.P.© – TeleTrusT Information Security Professional
V-Modell® XT

In addition, iSQI hosts seminars and organizes different international software conferences, for example the „Conference on Quality Engineering in Software Technology" (CONQUEST). It provides a platform for experts and practitioners from industry and science to exchange experiences and ideas and to discuss the newest advancements in their special field.

The CONQUEST conference program is included in this issue of testing experience.

www.isqi.org
SETTING QUALiTY STANDARDS
ALL OVER THE WORLD

thinking [Beizer], research on the TMM carried out in the EU-funded MB-TMM project, and international testing standards, e.g. IEEE 829 Standard for Software Test Documentation [IEEE 829]. The testing terminology used in the TMMi is derived from the ISTQB Standard Glossary of terms used in Software Testing [ISTQB].

The model is structured much like CMMi with 5 levels of maturity. At level 2 of the model projects begin to be assessed on their capability to measure progress through testing. However like the CMMi model it is not prescriptive as to what is measured just that measurement occurs, although help is provided regarding the minimum is expected.

Tools

Now to some these are the panacea, I prefer to look at tools as the slaves of the process. When it comes to information provision there does seem to have been thought put in most examples I have seen, to the data that the tool provides. The Dashboard seems to be the latest fashion accessory for any company wishing to set its stall in the test management arena. HP has one in Changepoint, Compuware in ODM etc, etc. In principle these provide data derived from actions within the system.

Tools can be excellent providers of data, but that data is only as good as the way it is collected. A common problem, for example, with Test Management tools is that the tool is implemented straight out of the box, with no configuration; testers then load test cases into the tool with no views on how progress will be tracked, so they load the minimum data to allow them to do their job. Then they try to track progress, but realise that the data needed to do this hasn't been loaded, so they can track certain elements, so unfortunately in my experience the reporting in these situations is very poor.

So with some real thought up front as to what information/data is required the tools can be configured to ensure that data is a by product of the process (e.g. does not become time consuming to collect and report).

Each approach listed above contributes in a small but very useful way in helping the Test Manager recognise the data that is required, and to some extent the way that data should be communicated, however this doesn't extend across the different roles that exist within a project. It doesn't prescribe what it is a development manager needs to know or the Business Sponsor.

So having identified the data, the key is how it is presented back as information in the right format at the right time.

This particular issue does seem to reside in the Test Managers camp, so I would like to focus there for now.

I believe that the future of testing depends on the following:-

A. Accurate information provision, in the language of the recipient
B. Building quality in, rather than testing it out, prevention rather than detection
C. Ensuring that quality is considered and not seen as a burden, alongside time and cost, which seem to have become the new buzzwords

As I have said already the great thing about Qualifications is that they have established some common terminology amongst testers, but that terminology is still not translatable across the lifecycle. So we have four languages as a minimum being spoken in projects today, we have the Business (client), the designers, the developers and the testers, and in an ITIL (IT Infrastructure Library) organisation is possibly a fifth.

We have projects that sit together but don't talk to each other. We also have so called Agile projects starting to show how greater collaboration brings big benefits, is this the way forward?

I think the way forward for information provision is actually to learn something from the Agile world and start to talk to each other, and not to work in silos. As I have already explained, everyone's requirement for information is different and so the closer projects work together and people mix the more we can understand what our requirements are.

In a world of increased outsourcing or offshoring this becomes increasingly harder to achieve. This is especially relevant when the test resources, or development resources, work across the other side of the world in a different time zone. It is as important, if not more important, that in these situations proper and regular communications are established. It's not easy but if success is expected then effort needs to be put in to ensure that each party is aware of their goals and provides the right information to enable measurement of progress to be tracked and understood.

Lots of organisations seem to think that by putting people in buildings in different countries and arming them just with a process they will get great results. This ignores the need to work as a team and communicate, and so in most instances is a strategy doomed to failure.

I think it worthy of a stop here to review another element we need to consider. We (and in this instance I am not just referring to test resources, I refer to anyone involved in a software development project) need to accept that Software lies at the heart of almost every business process. Everything from how an organisation communicates internally and externally, to the way it delivers products and services, to how it manages administrative processes. Software can liberate or debilitate an organisation's efficiency, effectiveness, risk exposure, risk exposure, and ultimately, its profitability.

So it's not something a Test Manager can do alone, we have to work together to ensure that in the increasingly complex world of software development we understand what it is we all need, and ensure we deliver it.

If the test manager for Terminal 5 had provided the following information to Willie Walsh at the point that the decision to reduce testing occurred I wonder if the result may have been different.

Test Manager –"Mr Walsh we have reviewed progress and remaining activity and can report that should you reduce testing in the way suggested the following will happen:

In the first 5 days of operation:

1. Over 23,000 bags will get lost, leaving you with a bill well in excess of £300,000 to reconnect them with passengers
2. 500 flights will definitely be cancelled
3. Staff will not be able to park due to issues with the security system, and so will not arrive on site on time
4. 28 lifts will not be operating
5. You will definitely be pulled up in front of the UK Parliament to explain yourself

I recommend we delay launch for a minimum of a week so we can resolve these issues."

Would he have agreed to the same launch date? That I cannot say, but at least if he had, he would have understood the issues he was to face with the press and public.

But you may say hindsight is a great thing. If we could all see into the future we could perhaps all determine these types of issues easily, but how do we do that prior to delivery?

Project delivery is not difficult, it is simply the bringing together of professionals in their field to deliver to a single goal. If that team works together then, for example, how much can a developer learn from a business analyst if they talk about how the system should actually work, or where it always doesn't!

A project manager who just cares about meeting delivery dates regardless of setting, and meeting success criteria is a fool in my book, and one that shouldn't be working in our industry. Success criteria are the goals that are the starting point for information; and may be different for each different level in the project, for example:

1. Success for the Sponsor may be a system that saves or makes them £3bn a year. In that instance the information provided needs to include data to reflect achievement when it goes live

2. Success for a Test Manager may be that there is an agreed risk based approach to testing established at the outset of the project, and all high priority test cases are agreed up front and are completed during

test execution, but let's think about that. So to be successful a Test Manager needs to have a prioritised list of test cases, which they ensure are completed during test execution.

This is an altogether too familiar success criterion today but maybe one that doesn't help point B above (building quality in, not testing it out), maybe a better success criteria could be:

a. Test activity defined that reducing the amount of test execution to one week (a tall order but if the team works together and identifies up front where things may go wrong and stop them occurring. Using an analogy, if you knew early enough that your wall was about to fall down, would you write a test and then go away until it fell? Then check it had, raise a bug report and pay a lot of money to have it repaired? Or would you do something as soon as possible to stop it falling and keep the cost, time and quality at the right levels? Translating that into a project, if we know the objectives of a project manager include time, cost and quality we need to find ways of preventing rather than detecting issues as this is the most economical approach to delivery, and to meet all three objectives).

Test managers and project managers must also reflect progress towards achieving these criteria. But we shouldn't forget that throughout the lifecycle of a project these will change as each new issue and risk is resolved, and so by necessity, must the information provision.

Teamwork is absolutely key to achieving project delivery in the future, and ensuring software testing continues to deliver value. Sadly this is something that gets very little daylight in IT. We ensure all of our people are technically competent but we don't really consider that they need to be taught how to manage people and how to build working teams. Good developers are promoted and asked to manage people, and it's the same in testing. How many testers ever get sent on a project management course to learn the basic project management processes? Very few, in fact in an organisation I worked in it was considered a course that was too advanced for test resources to attend! However developers and designers aspiring to become Project Managers were allowed to attend.

I think a software delivery project is much like an orchestra, if each part works together in harmony it can be beautiful, but if only one element works on their own the result will be disastrous. We need to work together so that the software delivery orchestra works, and delivers beautiful results.

So now let's go back to where I started. Good information provision comes from being part of a team who communicate regularly and fully understand the progress through the different lifecycle stages, all the way through to the live environment.

At this point I can't help but remember the quote – '"You cannot manage what you cannot (or do not) measure". (anon). So we must know:

- What do we need to measure and does the information and importance of the information change during the lifecycle of the project
- We must still capture the number of test cases, tests run, tests passed/failed, but need to identify how to translate this information for the different audiences we have
- We need to capture 'quality requirements' and therefore identify how to measure meeting them. They must be clearly and concisely defined, easily understood, not ambiguous and easily demonstrated
- Testers must encourage the breakdown of the silos. Why not work within the development team and help them test?

No one can define today what the future of testing is but by learning to communicate more effectively and by becoming an integrated element of the project, I believe the future of testing is assured.

Biography

Geoff is Consultancy Director of Experimentus (www.experimentus.com). He has been actively involved in testing for over 20 years covering Test Analysis, Test Management, Test Programme Management and Test Strategy Development.

Geoff is a trained Test Maturity Model (TMM) assessor and can perform Test Maturity assessments. Obtaining the accolade as Test Manager of the Year 2004, Geoff was also one of only three testers in Europe nominated for The European Testing Excellence Award in 2003.

Geoff is a recognised thought leader in his field. He is a founder member of the ISEB Software Testing Board, International Software Testing Qualifications Board (ISTQB) and the TMMi Foundation, of which he is currently Treasurer. Geoff was directly involved in the creation of the original ISEB Foundation syllabus in Software Testing in 1999, managed the delivery of the ISEB Practitioner syllabus in 2002 and is currently the UK representative to the ISTQB - formulating and managing the new Foundation, Advanced and Expert level syllabi.

Geoff is also Chairman of The UK Testing Board and Vice Chairman of the BCS SIGiST (Specialist Group in Software Testing).

Geoff's expertise is widely documented through articles and publications. He co-authored what is currently classified as 'the best selling testing book on Amazon.co.uk', ISEB – A Software Testing Foundation. Geoff is also a frequent speaker at major industry events nationally and internationally.

© iStockphoto

What Testing Cannot Do

by Gerald M. Weinberg

An excerpt from Chapter 2 of Gerald M. Weinberg's July 2008 release, *Perfect Software—And Other Illusions About Testing*, ISBN:978-0-932633-69-9. Reprinted by permission of Dorset House Publishing. To save 20% off the $23.95 list price before September 1, 2008, visit www.dorsethouse.com/offers/te.html, e-mail info@dorsethouse.com, or call (212) 620-4053.

"Have no fear of perfection—you'll never reach it."
—Salvador Dali, Spanish Surrealistic Painter (1904-1989)

Imagine you are an executive in charge of a multimillion-dollar logistics system. The company president, Benito, is a dynamic, forceful character known for firing people in staff meetings when they don't give him the answers he wants. He is eager to have the system rolled out. He's putting pressure on you. You go to the project manager, who reports to you, and ask, "How's progress on the system?" The project manager replies, "I don't know."
What would you do now? Quite likely, you'd want more information than "I don't know." There are various ways to acquire it:

a. Maybe torture will work.
Pull out the project manager's fingernails one at a time until she squeals, "Oh, yes, I do know. It's great!" If necessary, proceed to toenails.

b. Maybe promoting a toady will work.
Fire the project manager and promote someone who says, "I do know." Repeat this step as needed.

c. Maybe gathering information will work.
Look for some information on what the program actually does when someone tries to use it in a realistic way.

If you want reliable information, *a* and *b* probably won't do the job, but *c* at least has a chance. Gathering information on what a program actually does when used is one form of what some people call "testing."

Information doesn't necessarily help reduce risk.
Managers in the software business frequently have to make risky decisions, which can be made less dangerous if based on answers to questions such as the following:

- Do we ship now? Later? Ever?
- Do we cancel the project?
- Do we continue the project and add more resources?
- Do we reduce the scope of the project?
- Do we attempt to bring the product to a wider market?
- Do we sell this version of the product at a lower price?

Of course, such decisions could be made in the complete absence of information—and they frequently are. More often, they're made after the decision-maker has obtained some bit of information, while resolutely ignoring other information that might be obtained. Since testing is an information-gathering process, there's always a question of whether it's worthwhile to pay for more testing. And that is another decision.
Someone in the organization has the authority to make decisions, with or without information. If that person will make the same decision no matter what additional information arises, there is little point in testing to gather more information. If Benito has decided to go forward with the new logistics system as is, why bother getting him more information? Why bother testing?
In fact, sometimes doing more testing adds to the risk. If Benito delays the product while waiting for more tests on the logistics system, he might enter the market too late. Or, he might go broke spending money on testing. So, we always have to consider money and time in the decision about whether to test: Testing cannot be done in zero time and cannot be done for free.
Sometimes, the information produced also adds to the risk. If the developers have information that something isn't working well, they may want to spend time fixing it. From the point of view of Benito, this could increase the risk of spoiling something that was working well enough to use.
You might think the real risk is in having a development organization you can't control, but here's another danger. If people sue you because of an error in software you developed and sold, they can subpoena the records of your development process. If the records indicate that your testers found this bug and you didn't fix it, you're in more trouble than if they'd never found it in the first place. Sometimes, ignorance really is (legal) bliss.
The same blissful-ignorance principle translates down to the individual manager, even if no lawsuit is involved. If there's a problem with the product in the field and the manager can honestly say, "I didn't know," the repercussions usually are less severe than if that manager knew but didn't do anything about it. The moral: Think carefully about what may qualify as too much information. But think even more carefully about which of your goals are foremost. Are you trying to produce a successful product, or avoid lawsuits, or simply advance your personal career?

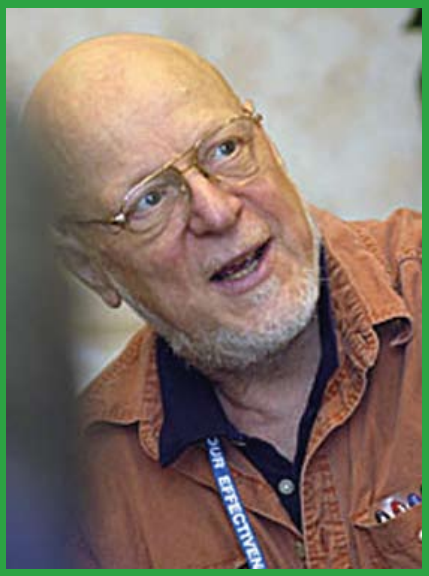


Photo by Steven M. Smith, www.stevenMsmith.com

Biography

Gerald Weinberg is the author of more than forty books spanning all phases of the software development life cycle. For nearly fifty years—through his writing, speaking, teaching, and consulting—Weinberg has helped generations of software professionals to improve their productivity. Based in New Mexico, he is a principal of Weinberg and Weinberg, consulting to consultants, start-ups, government agencies, and Fortune 500 firms. Visit www.geraldmweinberg.com to learn more.

APPLICATION SECURITY

www.diazhilterscheid.com

How high do you value your and your customers' data? Do your applications reflect this value accordingly? Accidental or deliberate manipulation of Data is something you can be protected against.

Talk to us about securing your Systems. We will assist you to incorporate security issues in your IT development, starting with your system goals, the processes in your firm or professional training for your staff.

as@diazhilterscheid.com

© iStockphoto

Co-Founder of ISSECO (International Secure Software Engineering Council)

Two Weeks to Better Testing this Summer

by Rex Black

Many of us will spend this summer relaxing, which is always good. However, why not invest a little of your vacation time in improving your testing operation. After all, if you're like most testers, you are time constrained and need to make improvements quickly that show fast results. So here are three practical ideas which you can put into action in just two weeks, which will make a noticeable difference when you tackle that next big, post-summer project.

Get Hip to Risk-Based Testing

I have a simple rule of thumb for test execution: Find the scary stuff first. How do we do this? Make smart guesses about where high-impact bugs are likely. How do we do that? Risk-based testing.

In a nutshell, risk-based testing consists of the following:

1. Identify specific risks to system quality.
2. Assess and assign the level of risk for each risk, based on likelihood (technical considerations) and impact (business considerations).
3. Allocate test effort and prioritize (sequence) test execution based on risk.
4. Revise the risk analysis at regular intervals in the project, including after testing the first build.

You can make this process as formal or as informal as necessary. We have helped clients get started doing risk-based testing in as little as one day, though one week is more typical. For more ideas on how, see my article, "Quality Risk Analysis," in the Library at www.rbcs-us.com, or my books *Managing the Testing Process* (for the test management perspective) or *Pragmatic Software Testing* (for the test analyst perspective).

Whip Those Bug Reports into Shape

One of the major deliverables for us as testers is the bug report. But, like Rodney Dangerfield, the bug report gets "no respect" in too many organizations. Just because we write them all the time doesn't mean they aren't critical—quite the contrary—and it doesn't mean we know how to write them well. Most test groups have opportunities to improve their bug reporting process.

When RBCS does test assessments for clients, we always look at the quality of the bug reports. We focus on three questions:

1. What is the percentage of rejected bug reports?
2. What is the percentage of duplicate bug reports?
3. Do all project stakeholder groups feel they are getting the information they need from the bug reports?

If the answer to questions one or two is, "More than 5%," we do further analysis as to why. (Hint: This isn't always a matter of tester competence, so don't assume it is.) If the answer to question three is, "No," then we spend time figuring out which project stakeholders are being overlooked or underserved. Recommendations in our assessment reports will include ways to gets these measures where they ought to be. Asking the stakeholders what they need from the bug reports is a great way to start—and to improve your relationships with your coworkers, too.

Read a Book on Testing

Most practicing testers have never read a book on testing. This is regrettable. We have a lot we can learn from each other in this field, but we have to reach out to gain that knowledge. (Lest you consider this suggestion self-serving, let me point out that writing technical books yields meager book royalties. In fact, on an hourly basis it's more lucrative to work bagging groceries at a supermarket. Other benefits, including the opportunity to improve our field, are what motivate most of us.)

There are many good books on testing out there now. Here's a small selection, any one of which you could work your way through during a beach vacation.

I have read each of these books (some of which I also wrote or co-wrote). I can promise you that, if you need to learn about the topic in the left column of the table, reading one of the books in the right column will repay you in hours and hours saved over the years, as well as teaching you at least one or two good ideas you can put in place immediately.

What You Want	Books to Read
General tips and techniques for test engineers	*Pragmatic Software Testing*, Rex Black *A Practitioner's Guide to Software Test Design*, Lee Copeland
Object-oriented testing	*Testing Object-Oriented Systems*, Robert Binder
Web testing	*The Web Testing Handbook*, Steve Splaine
Security testing	*Testing Web Security*, Steve Splaine *How to Break Software Security*, James Whittaker
Dynamic test strategies and techniques	*T-Map Next*, Tim Koomen et al *How to Break Software*, James Whittaker
Test management	*Managing the Testing Process*, Rex Black *Systematic Software Testing*, Rick Craig
Test process assessment and improvement	*Critical Testing Processes*, Rex Black *Test Process Improvement*, Martin Pol et al
ISTQB tester certification	*Foundations of Software Testing*, Rex Black et al *The Testing Practitioner*, ed. Erik van Veenendaal

Biography

With a quarter-century of software and systems engineering experience, Rex Black is President of RBCS (www.rbcs-us.com), a leader in software, hardware, and systems testing. For over a dozen years, RBCS has delivered services in consulting, outsourcing and training for software and hardware testing. RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. As the leader of RBCS, Rex is the most prolific author practicing in the field of software testing today, having written four popular books, with two new books (and a new edition of Managing the Testing Process) on the way. Rex is the President of the International Software Testing Qualifications Board and a Director of the American Software Testing Qualifications Board.

The Boundary Value Fallacy

by René Tuinhout

Introduction

Boundary Value Analysis is a popular test technique to use: It is easy to comprehend and easy to explain to testers and non-testers alike.

Part of the ease in comprehension is based on the elegant and simple examples that can be used to explain the technique: A boundary value is introduced, and it's explained that three values are needed to test the boundaries thoroughly.

This article suggests that three values are, contrary to popular believe, not enough to perform a black box test to prove that a boundary has been implemented correctly. Furthermore, the article introduces a new technique, B³VA, to solve this issue in Boundary Value Analysis.

Equivalence Partitioning

The Boundary Value Analysis (BVA) test technique is based on the Equivalence Partitioning (EP) test technique. The basic idea of EP is quite simple:"If you expect the same result from two tests, it's sufficient to run one of them"[1]. This idea is based on two assumptions:

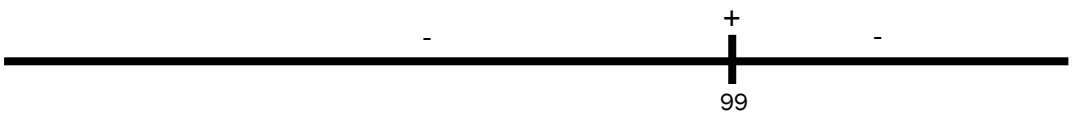
1. Every system input (or system output) can be divided into several classes;
2. The system processes every possible input value in a class in exactly the same way as it processes every other input value in that class.
   (or: the system has produced every possible output value in a class in exactly the same way as every other possible output value in that class has been produced.)

Based on these two assumptions it follows that it's sufficient to test one value per class[2]:

An example:
An insurance system accepts clients aged 99 years old or younger. All other potential clients are not accepted. Ages are entered into the system under test in whole years[3].

The above functional description leads to two classes:
1. People aged 99 and younger will be accepted (this is called a valid class);
2. People aged 100 and older will not be accepted (this is called an invalid class);

This can be represented in a figure (+ means: valid, - means: invalid):



When using EP, the system in the example can be tested by using (for example) the values:
15 (expected result: client accepted) and 120 (expected result: client not accepted).

(These values are arbitrarily chosen, just as long as for each class a value is tested. Therefore, also the values 89 and 160, or for that matter 20 and 200 could have been chosen.)
Both classes (the valid and the invalid class) have now been tested.

[1] Essential Software Test Design, Torbjörn Ryber 2007.
[2] In the following examples and elaborations EP based on input values will be used. However, the examples and elaborations can be extrapolated for EP based on output values.
[3] To keep the explanation as simple as possible, other obvious boundaries (no ages below zero, no alphanumeric input, no non-integer input, etc.) have been omitted in the example.

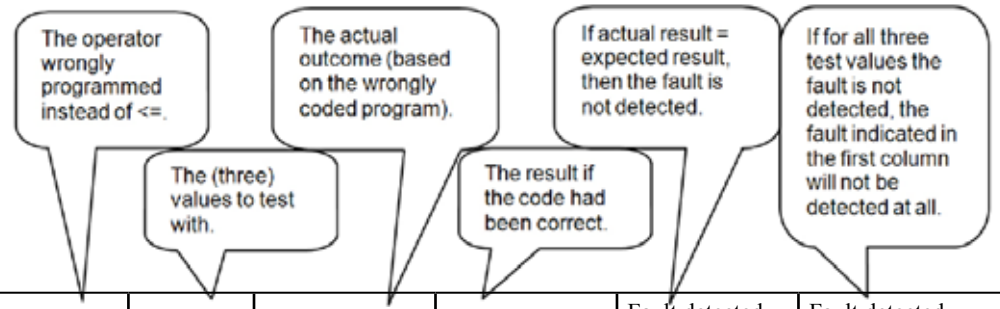
Boundary Value Analysis

While executing tests it was discovered some defects seemed to cluster around the values outlining the equivalence classes. It's not hard to see how this came about: The values outlining the equivalence classes (the boundaries) are being incorporated into the programming code during programming. In the example above, focusing on the "younger than or equal to 99 years old"-boundary, somewhere in the code it could say:"IF age <= 99".
Since this boundary is actually in the code, this is where a programming error could most easily occur. Therefore this is also the place to test the most thoroughly. In the Boundary Value Analysis test technique this means to test with test values on, just below, and just above the boundary to detect any possible programming fault made in the relational operator (<, <=, =, >, >=, <>). The use of three values is based on the following:

In our example, that could be written down in pseudo code as:

```
IF age <=99
 THEN accept client
 ELSE refuse client
END IF
```

the defects that could exist regarding the relational operator are programming faults in the <= sign. This operator could wrongly be programmed as: >, >=, <, <> or = instead of <=[4,5].



Possible fault	Test value	Actual Result	Expected Result	Fault detected per value	Fault detected using three values
> 99	98	Refuse	Accept	Yes	Yes
	99	Refuse	Accept	Yes	
	100	Accept	Refuse	Yes	
>= 99	98	Refuse	Accept	Yes	Yes
	99	Accept	Accept	No	
	100	Accept	Refuse	Yes	
< 99	98	Accept	Accept	No	Yes
	99	Refuse	Accept	Yes	
	100	Refuse	Refuse	No	
= 99	98	Refuse	Accept	Yes	Yes
	99	Accept	Accept	No	
	100	Refuse	Refuse	No	
<> 99	98	Accept	Accept	No	Yes
	99	Refuse	Accept	Yes	
	100	Accept	Refuse	Yes	

The table above shows that a programming fault in the relational operator will always be detected when using three test values (on, just below and just above the boundary). It also shows some programming faults will not be detected when testing using only two values[6].

BVA summary

In summary, the use of BVA comes down to:

1. Identify equivalence classes[7]
2. Determine Boundary Values
3. Specify test cases using three test values for each boundary.

The Boundary Value Fallacy

However, is the above correct? Are three values really enough to test if a failure occurs as a result of an incorrectly programmed relational operator? The evidence presented in the table seems to support it. Also, quite a lot of literature shows that testing using BVA should be done using three values[8].

So, is testing with three values really enough? To answer this question, we should find out if BVA is a white box or a black box testing technique. ISTQB defines BVA as:"A black box test design technique in which test cases are designed based on boundary values"[9].
BS7925-2[10] describes itself as a "Standard for Software Component Testing", and defines itself to "cover only the lowest level of independently testable software". It identifies black box techniques, but it doesn't identify BVA as being such a technique (nor does it identify BVA as being a white box technique).
In daily practice, BVA is used for white box as well as black box testing.

And that's where a possible issue occurs:
Let's have another look at the example presented earlier:

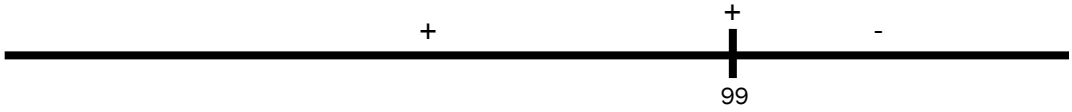
[4] The incorrect introduction of a "NOT" is not included in the possible defects, since this is covered in the defects mentioned (e.g. "NOT <= 99" equals ">99").
[5] Of course, other defects could be introduced, e.g. typing "999" instead of "99", typing "99z" instead of "99" etc. These defects will, however, not be discussed since boundary value analysis focuses on the possible defects mentioned in the text. (To test for the other defects presented in this footnote, other test techniques could be used.)
[6] E.g. If "= 99" was programmed instead of "<= 99", this fault would not have been detected testing only with the values 99 and 100 (since the actual outcome (produced by the code with the fault in it) would be the same as the expected outcome). The same goes for the incorrect programming of "< 99" instead of "<= 99". This programming error would not be detected if the test was executed using only the values 98 and 100.
[7] The way in which the identified equivalence classes should be represented (in a picture, a table or even using another technique) is not relevant for this article.
[8] Please refer to the Literature reference.
[9] Standard Glossary of terms used in Software Testing, Version 2.0 (dd. December 2nd 2007), Produced by the 'Glossary Working Party' International Software Testing Qualifications Board.
[10] Standard for Software Component Testing, Working Draft 3.4, Date: 27 April 2001 produced by the British Computer Society Special Interest Group in Software Testing (BCS SIGIST).

An insurance system accepts clients aged 99 years old or younger. All other potential clients are not accepted. Ages are entered into the system under test in whole years[11].

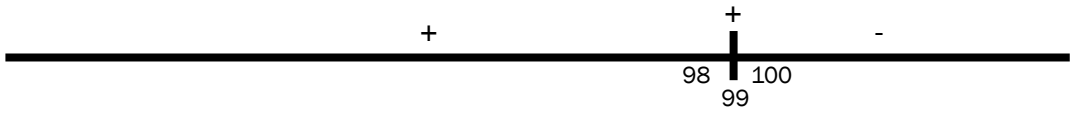
The above functional description leads to two classes:
1. People aged 99 and younger will be accepted (this is called a valid class);
2. People aged 100 and older will not be accepted (this is called an invalid class);

This can be represented in a figure (+ means: valid, - means: invalid):

Equivalence Classes:



BVA test values:



When contemplating this example, it occurs that 99 being the boundary value is based on an assumption. The assumption being that the given functional description ("people aged 99 and younger will be accepted, people older than 99 will not be accepted") is implemented as:

```
IF age <=99
 THEN accept
client
 ELSE refuse
client
END IF
```

However, this need not be the case. The functionality could as well be implemented as:

```
IF age < 100
 THEN accept
client
 ELSE refuse
client
END IF[12]
```

In black box testing, for which BVA is often used, the former implementation is as good as the latter: Since it is black box testing, the test professional has no insight into the implementation of the functional design.

However, does BVA still find possible faults in the relational operators if the latter implementa-tion is chosen by the programmer? Does it detect a fault if >, >=, <=, = or <> 100 is programmed instead of < 100?
Let's have a look:

Based on the functional design (being the test basis for a black box test) which states "An insurance system accepts clients aged 99 years old or younger and refuses to accept clients older than 99 years old", 99 will be identified as the boundary in a black box test[13]. The three values to test using the BVA technique will therefore be 98, 99 and 100:

This leads to the following table when using BVA with three values:

Possible fault	Test value	Actual Result	Expected Result	Fault detected per value	Fault detected using three values
> 100	98	Refuse	Accept	Yes	Yes
	99	Refuse	Accept	Yes	
	100	Refuse	Refuse	No	
>= 100	98	Refuse	Accept	Yes	Yes
	99	Refuse	Accept	Yes	
	100	Accept	Refuse	Yes	
<= 100	98	Accept	Accept	No	Yes
	99	Accept	Accept	No	
	100	Accept	Refuse	Yes	
= 100	98	Refuse	Accept	Yes	Yes
	99	Refuse	Accept	Yes	
	100	Accept	Refuse	Yes	
<> 100	98	Accept	Accept	No	No
	99	Accept	Accept	No	
	100	Refuse	Refuse	No	

[11] To keep the example as simple as possible, other obvious boundaries (no ages below zero, no alphanumeric inputs, no non-integer inputs, etc.) have for clarity, again, been omitted in the example.
[12] Again, assuming the input being integer has been thoroughly tested and implemented elsewhere in the code.
[13] Even if "If age < 100" is programmed. In black box testing, the tester is unaware of the implementation; hence the functional design is accepted as leading. The functional design mentions 99, therefore 99 will be identified as the boundary.

As shown in the table above, if "<> 100" was (wrongly) programmed instead of the intended "< 100", the actual results and the expected results are identical for all three BVA test cases. Therefore, the fault will not be detected!

Where does this flaw in BVA occur?

Based on the above it can be concluded that using three test values in BVA will, for black box testing, not always be enough to find a possible fault in the relational operators used. As shown, the flaw could lead to a wrong test result if "<" was intended to be programmed, and "<>" was actually programmed. However, there are more situations in which the flaw occurs[14,15]:

- When the black box test basis states "Input less than or equal to X"
- When the black box test basis states "Input greater than X"
- When the black box test basis states "Input greater than or equal to X"
- When the black box test basis states "Input less than X".

A remedy

In the example, the non-detection of the fault can be easily remedied by adding an extra test value (101), as shown in this table:

Possible fault	Test value	Actual Result	Expected Result	Fault detected per value	Fault detected using three values
> 100	98	Refuse	Accept	Yes	Yes
	99	Refuse	Accept	Yes	
	100	Refuse	Refuse	No	
	101	Accept	Refuse	Yes	
>= 100	98	Refuse	Accept	Yes	Yes
	99	Refuse	Accept	Yes	
	100	Accept	Refuse	Yes	
	101	Accept	Refuse	Yes	
<= 100	98	Accept	Accept	No	Yes
	99	Accept	Accept	No	
	100	Accept	Refuse	Yes	
	101	Refuse	Refuse	No	
= 100	98	Refuse	Accept	Yes	Yes
	99	Refuse	Accept	Yes	
	100	Accept	Refuse	Yes	
	101	Refuse	Refuse	No	
<> 100	98	Accept	Accept	No	Yes
	99	Accept	Accept	No	
	100	Refuse	Refuse	No	
	101	Accept	Refuse	Yes	

By adding the extra test value (101) the fault ("<>" being programmed instead of "<") will, even when using Boundary Value analysis as a black box technique, be found (in the table above this is shown for the 99-boundary value used in the example presented earlier).

To extrapolate this example to an overall solution, for every situation in which the fault occurs the tables below suggest a solution.

When FD states:	This could be implemented as:	But also as:	Fault will not be detected using BVA if ... is programmed	Instead of:	Fault will be detected when testing with extra value:
"If input less than or equal to X"	IF input <= X	IF input < (X+1)	<> (X+1)	< (X+1)	(X+2)
"If input greater than X"	IF input > X	IF input >= (X+1)	= (X+1)	>= (X+1)	(X+2)
"If input greater than or equal to X"	IF input >= X	IF input > (X-1)	<> (X-1)	> (X-1)	(X-2)
"IF input less than X"	IF input <= X	IF input <= (X-1)	= (X-1)	<= (X-1)	(X-2)

In an example:

When FD states:	This could be implemented as:	But also as:	Fault will not be detected using BVA if ... is programmed	Instead of:	Fault will be detected when testing with extra value:
"If input less than or equal to 99"	IF input <= 99	IF input < 100	<> 100	< 100	101
"If input greater than 99"	IF input > 99	IF input >= 100	= 100	>= 100	101
"If input greater than or equal to 99"	IF input >= 99	IF input > 98	<> 98	> 98	97
"IF input less than 99"	IF input <= 99	IF input <= 98	= 98	<= 98	97

Please be aware that the (X+1), (X+2), (X-1) and (X-2) in the table below should be interpreted as:

X+1: The closest value on the upper side of X with regard to accuracy and type of the boundary value X.

X+2: The closest value on the upper side of (X+1) with regard to the accuracy and type of the boundary value X.

X-1: The closest value on the lower side of X with regard to the accuracy and type of the boundary value X.

X-2: The closest value on the lower side of (X-1) with regard to the accuracy and type of the boundary value X.

FD: Functional design. "Fault will not be detected" means "Fault will not be detected if using BVA (3 values)"

A solution: B³VA

Now it's not only possible to conclude there's a flaw in BVA when BVA is used in black box testing, but based on the tables above a solution can be presented:

The BVA-technique can, for black box testing, be extended to Black Box BVA (B³VA):

1. Identify equivalence classes[16].
2. Determine test values using the Boundary Value technique.
3. Determine the possible operators used:

When the black box design states[17]	In formula	Then add an extra value next to the value on the	In formula	All test values overview
"Less than or equal to X"	<= X	upper side of the boundary	X + 2	X-1,X,X+1,X+2
"Greater than X"	> X	upper side of the boundary	X + 2	X-1,X,X+1,X+2
"Greater than or equal to X"	>= X	lower side of the boundary	X - 2	X-2,X-1,X,X+1
"Less than X"	< X	lower side of the boundary	X - 2	X-2,X-1,X,X+1

   If none of the phrases represented in this table is used in the design, no extra value needs to be added (all test values then being X-1, X, X+1).
4. Specify test cases, using three or four test values per boundary, based on the table in 3.

Please be sure to add possible extra values in the same way boundary values are found, and to interpret the X, X+1, X+2, X-1 and X-2 –values in the table correctly: With regard to the accuracy and type of the boundary value.

E.g.: In the example used before the value 101 is added as an extra upper value because:

- 99 is the boundary value (X);
- Integers (non-decimal numerics) are used.

The closest integer to the upper side of 99 therefore is 100 (X+1). The closest integer to the upper side of 100 then is 101 (X+2).

Should one decimal have been accepted in our example, then:

- 99 is still the boundary value (X); however

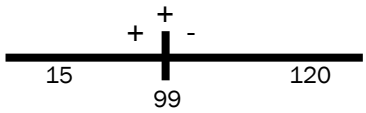
The closest value to the upper side of 99 would then have been 99.1 (X+1).

The closest value to the upper side of 99.1 would have been 99.2 (X+2).

If this rule of thumb is applied to the example used earlier, it shows the following.

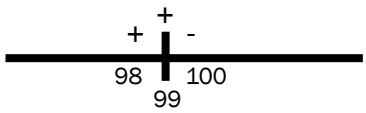
To test the 99-boundary introduced in the example:

1. EP:



Two test values, 15 (to test the valid class) and 120 (to test the invalid class), are chosen as test values. The boundary value (99) is not chosen as a test value[22].

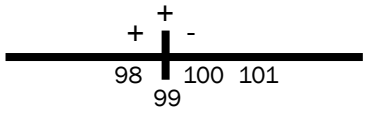
2. BVA:



Three test values are chosen:

- Two values in the valid class (98, 99) are chosen as test values
- One value in the invalid class (100) is chosen as test value[23].

3. B³VA:



Four test values are chosen:

- Two values in the valid class (98, 99) are chosen as test values, as in BVA
- One value in the invalid class (100) is chosen as test value, as in BVA
- One extra value in the invalid class (101) is chosen as test value[24].

Rule of thumb

When considering the B³VA technique as presented above, an elegant rule of thumb springs to mind. For this rule of thumb, please be aware that a boundary value always separates two equivalence classes[18, 19], (the boundary value itself being in one of the two classes).

Keeping this in mind and considering this article, it follows that for each boundary value:

1. When testing using Equivalence Partitioning (EP):
   One (random) value for each of the two equivalence classes separated by a boundary value must be chosen as test value;
2. When testing using Boundary Value Analysis (BVA):
   - One test value must be chosen (nearest to the boundary) in one of the two equivalence classes[20].
   - Two test values must be chosen (one on and one nearest to the boundary) in the other of the two equivalence classes[20]
3. When testing using Black Box Boundary Value Analysis (B³VA):
   - Two test values must be chosen (both nearest to but not on the boundary) in one of the two equivalence classes20.
   - Two test values must be chosen (one on and one nearest to the boundary) in the other of the two equivalence classes[20].

[16] The way in which the identified equivalence classes should be represented (in a picture, a table or even using another technique) is not relevant for this article.

[17] In any form

[18] All examples so far have shown a boundary value separating a valid and an invalid class. However, it can be shown the described also holds when the boundary value separates two valid (or two invalid) classes.

[19] Please be aware even a boundary value that is a class in itself (e.g. "customers are only accepted if they are 99 years old) can be seen as separating two classes (but twice: The two classes "99" (+) and "< 99" (-) are separated by the boundary value. Also the two classes "99" (+) and "> 99" (-) are separated by the boundary value).

[20] Referring to the two equivalence classes being separated by the boundary value. A boundary value that is a class in itself does not occur for B3VA.

[21] If the boundary value is an equivalence class in itself, then values must of course be chosen as follows: 1 on the boundary, 1 in the class on the left of and 1 in the class on the right of the boundary, but the latter two nearest to the boundary.

[22] It could have been chosen though. In that case the valid class would have been tested using the 99-value, dismissing the need to use 15 as a test value (because the valid class in that case is already tested using the 99-value).

[23] The spread of values over the classes is due to the example. In another example, the spread might have been diffe¬rent. (E.g. If "If < 99" was to be (white box) tested using BVA, 99 would have been in the invalid class. Still, 98, 99 and 100 would have been chosen as test values, thus testing one valid class value and two invalid class values.)

[24] In every (!) example using B3VA, two values for each class would have been chosen. E.g. in the example presented in footnote 26: if "Accept if age less than 99 years old" was to be (black box) tested using B3VA, the extra test value to choose using B3VA would have been 97. The boundary test value 99 would have been in the invalid class. Thus, 97 and 98 would have been tested, being two values in the valid class, and 99 and 100 would have been tested, being two values in the invalid class.

Summary and conclusion

In this article the Equivalence Partitioning technique (EP) and the Boundary Value Analysis technique were briefly described, and a BVA shortcoming was presented: The presented flaw in BVA can lead to the possible non-detection of a fault, and thus to extra risk, if the (black box) test basis contains functional descriptions in the form:

- "… should be less than (or equal to) …" or
- "should be greater than (or equal to) …".

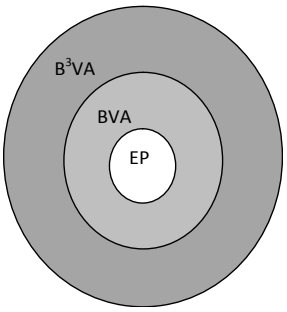
Subsequently, a solution for this shortcoming, the Black Box Boundary Value Analysis (B^3VA), was introduced and rules for using this technique were presented. It was shown that, by introducing an extra test value adhering to the B^3VA rules, the BVA-flaw is solved by using B^3VA.

B^3VA and Risk-based Testing:

Furthermore, one could argue that knowledge about EP, BVA and B^3VA can be used in test projects to make a well-founded choice regarding test coverge for black box testing:

- Choose EP if the only issue that needs to be covered is the correct functioning of the system under test with respect to its classes, regardless of the boundaries;
- Choose BVA if not only the correct functioning of classes has to be proven, but also a certain amount of certainty (not 100%!) needs to be established that the system under test functions correctly around and on its boundary values;
- Choose B^3VA if not only the correct functioning of classes has to be proven, but also certainty needs to be established that the system under test functions correctly around and on its boundary values.

Testing a system under test using BVA implicitly tests this system for EP.
In the same way, testing a system under test using B^3VA implicitly tests this system for BVA.
In a picture:



This enables the tester/test manager to not only choose an appropriate technique for a (black box) test, but also to allocate test priorities.

For example: In the example used in this article ("a person less than or equal to 99 years old is accepted for an insurance, persons older than 99 years are not accepted"), the tester/test manager could choose to make the:

- 98 and 100 test values:
  to be "Must test" test values (thus covering EP);
- 99 test value:
  to be a "Should test" test value (thus covering BVA);
- 101 test value:
  to be a "Could test" test value (thus covering B^3VA).

based on the risks involved[25], thus deciding to execute a certain test value based on the risk accepted by management, the time available, test progress etc.

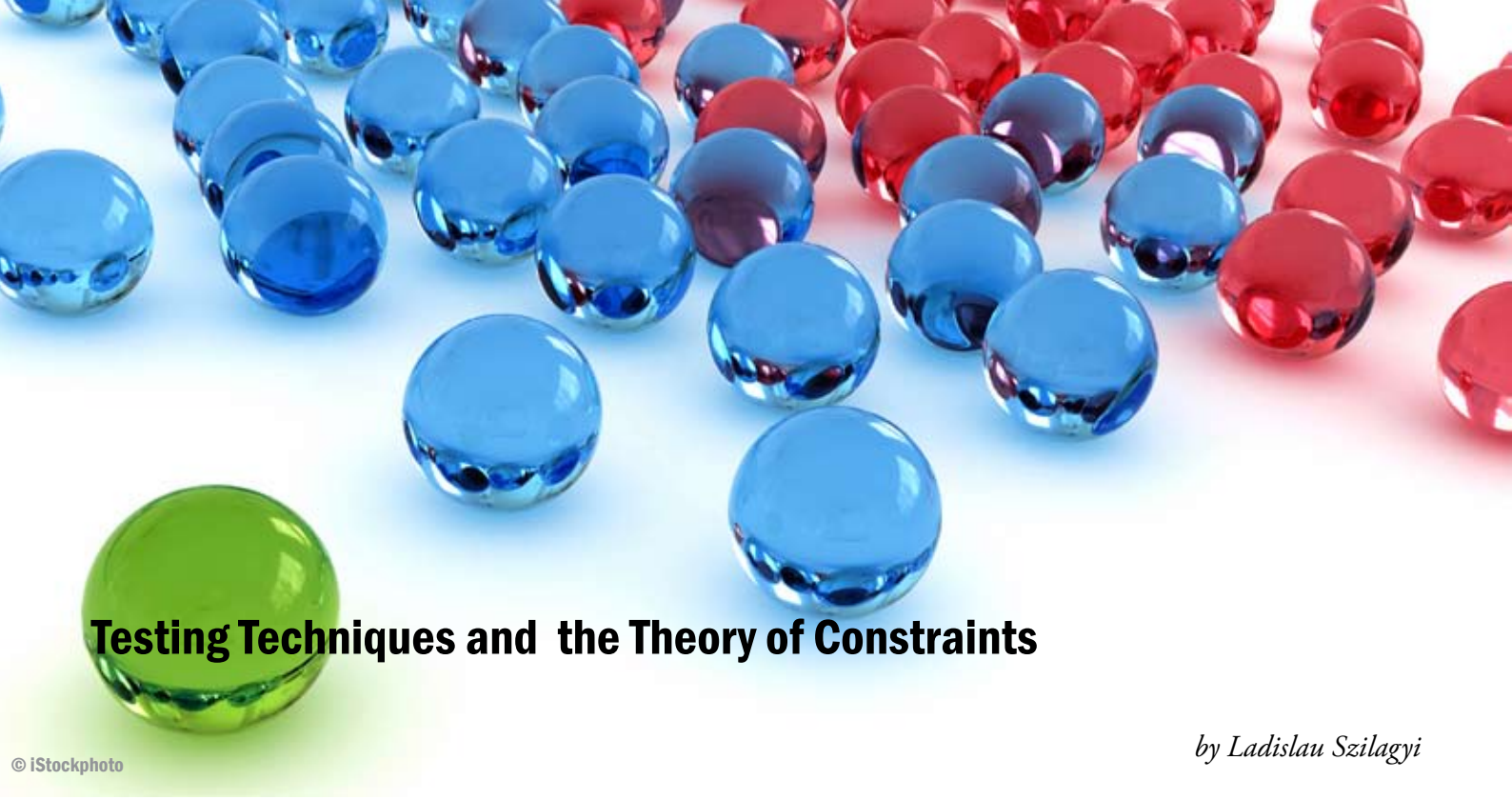
Literature

- *British Standard 7925-2, Standard for Software Component Testing, Working Draft 3.4, 27 April 2001*
  This Standard uses three test values per boundary. All examples given are white box, exactly within the scope described in the standard:"…so the standard is deliberately limited in scope to cover only the lowest level of independently testable software. Because the interpretation of and name for the lowest level is imprecise, the term 'component' has been chosen rather than other common synonyms such as 'unit', 'module' or 'program' to avoid confusion with these more common terms and remain compatible with them."
  The standard (therefore) does not elaborate on BVA black box testing.
- *Essential Software Test Design, Torbjörn Ryber, Mediaprint Uddevalla AB, Sverige 2007*
  This book deals with "design techniques … most important for functional tests" and explains "…this section builds on the British Standard 7925:2 for component testing". The book explains BVA in a short way and suggests in its text and tables that two test values per boundary are enough. However, in a graphical description of BVA for one of the two borders in the figure, three test values are suggested.
  However, the risk involved in BVA black box testing, or the need for four test values in some (black box) circumstances is not identified.
- *Practical Software Testing: a process oriented approach, I. Burnstein, Springer-Verlag New York.*
  In this book BVA is introduced as an extra to EP, it states "if a tester is viewing the software under test as a black box with well defined inputs and outputs, a good approach to selecting test inputs is to use a method called equivalence partitioning", thus defining EP (and with that also BVA) as a black box technique. For testing ranges using only two values per boundary is suggested. For strict boundaries, in an example three test values per boundary are presented (just below a boundary, on a boundary, just above a boundary). However, it is suggested later on that test values in the same class as the boundary itself could be omitted "if the tester does not believe they are necessary". The book stresses the need for equivalence partitioning and boundary value analysis to "be complimented by use of white box and, in many cases, other black box test design approaches." The risk involved in BVA black box testing, or the need for four test values in some (black box) circumstances is not identified.
- *TMap Next, Tim Koomen et. al., Tutein Nolthenius 2006.*
  This book distinguishes between "normal BVA" (testing using three values) and "BVA light" (testing using only two values). It does indicate the risk involved in testing with only two values.
  However, it does not identify the risk involved in BVA black box testing, nor does it identify the need for four test values in some (black box) circumstances.

Biography

René Tuinhout, Logica, is a senior test advisor and test coach with over 11 years of experience in software testing. René advises organizations on test management, test change management, structured testing, and he manages test (change) projects.
René is an ISTQB Practitioner and tutors courses in Test Techniques, ISTQB Foundation Level and several other test-related subjects.

[25] Must, Should and Could are chosen arbitrarily in this example, but should in real life be based on a product risk assessment, of course.



Testing Techniques and the Theory of Constraints

© iStockphoto

by Ladislau Szilagyi

Introduction

In the context of newly emerging testing schools and methodologies (see *Derk-Jan de Grood's Test Goal - Result driven testing*[4]), the topics of *Test strategy, tactics, testing mission* are hot and much debated in the worldwide testing community.

Also, in a real software project's day-to-day context, there are always constraints affecting the testing goals' optimal fulfillment.
How do we deal with these constraints? Here, we must use our **testing techniques** practical skills! Appropriate testing techniques must be used as mandatory key elements in this process of dealing with testing constraints, and *Eliyahu Goldratt's* way of thinking in ***Theory of Constraints*** may provide us with a proven methodology to be applied.

What is a testing process constraint?

According to Webster's dictionary, a constraint is "the state of being checked, restricted, or compelled to avoid or perform some action". I prefer to describe a constraint as "the conflict (between two actions), affecting the optimal fulfillment of an intended goal".

An example will help! Consider the following common situation: We are asked to perform *efficient and effective testing*. Apparently, that means we should obtain high test coverage in order to find as many bugs as we can (effective) and we should minimize test time (efficient). In order to obtain high test coverage, we should *execute as many test cases as we can*, whilst at the same time, to minimize test time, *we* should *execute only some of the test cases* (fig.1). That's a testing process constraint; We can't have both!

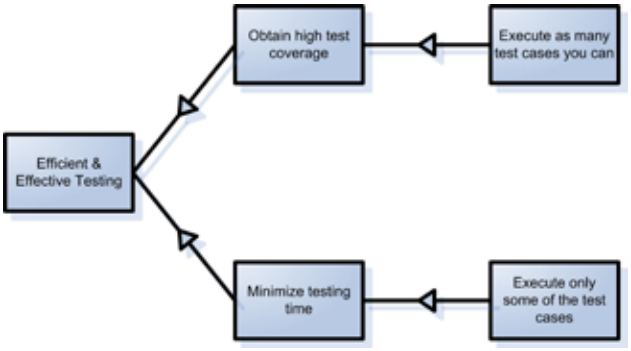


Fig 1. Constraint example

How can we deal with constraints? Here, we must use our **testing techniques** practical skills! It is obvious that the mastery of various testing techniques can help us find a way out of such situations. New testing techniques must be used as mandatory key elements in this process of dealing with testing process constraints. However, without an appropriate methodology, we may randomly jump to using another (possibly unsuitable) testing technique, and precious time will be wasted. I suggest that the thinking mechanism behind ***Theory of Constraints*** may guide us as a proven methodology.

The Theory of Constraints

The Theory of Constraints (TOC) is an overall management philosophy. It is based on the application of scientific principles and logic reasoning to guide human-based organizations. TOC is one of the favorite buzz-words of the current business management strategy school.

Obviously, it is out of the scope of this article to offer a detailed description and explanation of TOC. Instead, we will try to cover only some of the main ideas behind this theory.

In his bestseller "*What is this thing called THEORY OF CONSTRAINTS and how should it be implemented?*"[1], Eliyahu Goldratt, the author of TOC, points out the essence of his theory:

1. **Identify the Constraints**. Remember that to identify the constraints also means to prioritize them according to their impact on the goal.
2. **Decide How to Exploit the Constraints**. Now that we decided, how we are going to manage the constraints, and how should we manage the vast majority of the system's resources which are not constraints? We should manage them so that everything that the constraints are going to consume will be supplied by the non-constraints.
3. **Subordinate Everything Else to the Above Decision**. It's obvious we still have room for much more improvement. Whatever the constraints are, there must be a way to reduce their limiting impact.
4. **Elevate the Constraints**. But, can we stop here? There will be another constraint, but if we continually elevate a constraint there must come a time when we break it. Whatever we have elevated will no longer be limiting the system. Will the system's performance now go to infinity? Certainly not. Another constraint will limit its performance and thus…
5. **Go Back to Step1.**

TOC Thinking Processes

The Thinking Processes emerged as TOC practitioners needed to identify the core con-

straints and how to manage or elevate them. They needed the answers to three simple questions:

- What to change?
- What to change to?
- How to cause the change?

The thinking process steps of TOC uses some exotic terms:

1. Define the system – build the Intermediate Objectives Map
2. Analyze the mismatches - build the Current Reality Tree
3. Create a transformation – applying the "Evaporating Clouds" method
4. Design the future – build the Future Reality Tree
5. Plan the execution – build Prerequisite Trees

The decisive step to obtain the conflict resolution is the so-called "Evaporating Clouds" method (in fact, a method to "evaporate" conflict). Let's try to detail the underlying thinking process,by taking a look under the hood of the TOC Thinking Processes!

Evaporating Clouds

A "Cloud" is based on *Necessary Condition Thinking*: we must "execute as many test cases as we can" (wanted) as a necessary condition to obtain "high test coverage" (needed). We have already seen that when two "wants" appear to be mutually exclusive, there is a conflict. To solve this conflict, the way forward is to recognize that our "wants" are motivated by underlying "needs". We "want" to execute as many test cases as we can because we "need" to obtain high test coverage.

Let's build a graphic representation of our Conflict (fig.2).

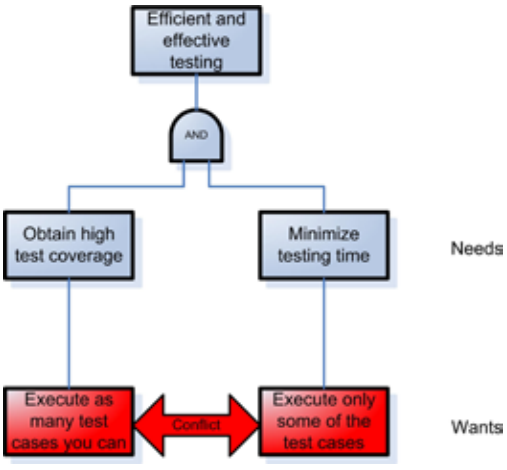


Fig 2. Needs and wants

The steps to follow in order to apply the "Evaporating Clouds" method are:

1. Identify the Wants ("execute as many test cases as you can", "execute only the most important test cases")
2. Identify the Conflict (can't have both!)
3. Identify the underlying Needs ("obtain high test coverage", "minimize testing time")
4. Identify the common Objective ("efficient and effective testing")
5. Identify and validate the Assumptions (in order to satisfy the Need we must obtain our Want because of our Assumptions)
6. Propose Solutions

The assumptions are basically "why" we must obtain our Want, and finding erroneous assumptions is the key to breaking the conflict. Assumptions "hide" underneath the Want-to-Need edges, and the Needs-to-Common Objective edges. There are also assumptions that underlie the Conflict entity itself— why we believe we can't have both Wants simultaneously.

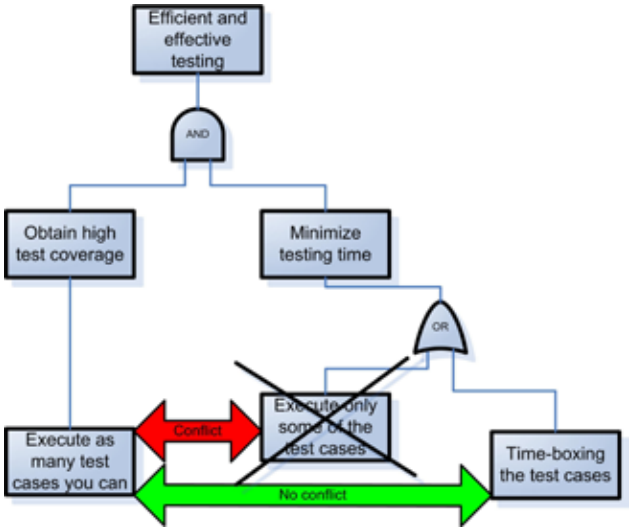


Fig 3. Propose Solution

Let's target the "execute only the most important test cases" Want. We assumed that in order to "minimize testing time" we *must* "execute only the most important test cases". Well, *must* or *may* execute? What if we imagine that perhaps there is another way to minimize testing time? Here, our testing skills shall guide us to search for alternatives…and one alternative is the risk-based testing approach! Yes, we could make a quality risk analysis (see5), prioritize the product features and come up with testing tactics involving various levels of testing (extensive, balanced, according to opportunity) and implying different time slices assigned to the test cases, depending on the risk of the feature being verified ("time-boxing the test cases" fig.3).

Oops, we just got our Solution, it satisfies our need to minimize the testing time (fig.4)!

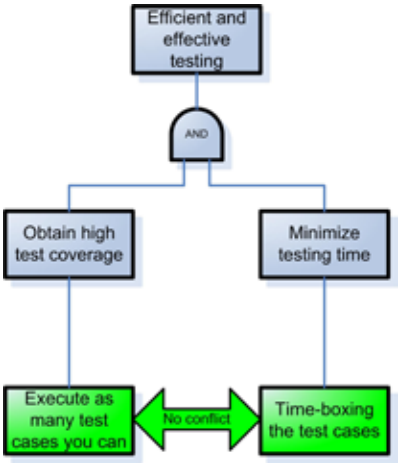


Fig 4. "Evaporated" conflict

Strategy, tactics and the practical use of TOC to remove testing process constraints

When we discuss the testing process, it's best to consider the whole picture. In the test process, the following levels define and apply test activities:

1. Test Plan (scope, approach, resources and schedule of intended test activities)
2. Test Strategy (high-level description of the test levels to be performed and the testing within those levels)
3. Test Tactics (practical actions performed by the tester when testing)

Strategy and tactics are different entities and are planned and performed in distinct time intervals. Strategy is setting the goal. In other words, the strategy sets the "What for?". There is no such thing as a 'unique test goal', but obviously there is a primary goal and there are many secondary testing goals, more exactly represented as a tree of inter-dependent testing goals (fig.5).

Tactics, on the other hand, are supposed to tell us "How we are supposed to reach the objectives." In other words, tactics answers the "How?". For any meaningful testing action we should be able to ask: "Why are we doing this? What is its purpose?". The answer to these questions is the strategic goal. Likewise, for any meaningful testing strategic goal we should be able to answer: "How do we obtain this? What actions are needed in order to achieve it?". The answer to these questions is the tactic action. That means that there must be

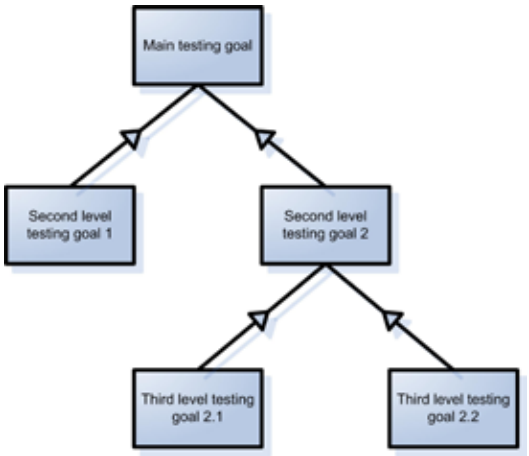


Fig 5. Testing goals tree

a corresponding tactic action behind any strategic goal, (fig.6).

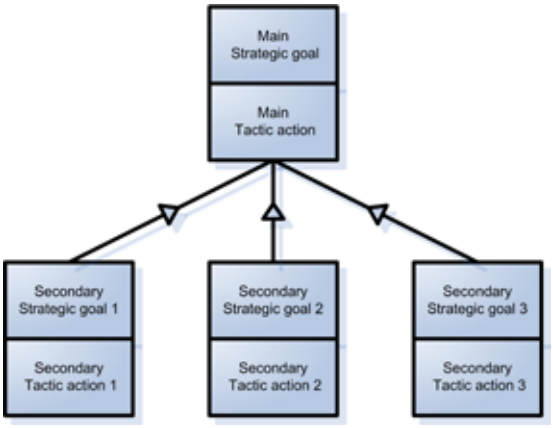


Fig 6. Strategic testing goals and tactic actions tree

Usually we cannot modify the test strategy *during the testing session*. The right time to update it will be *after* the testing session, when we learn the lessons (success or failure) from the testing mission. However, if we are doing *Exploratory testing*, we *may* modify our strategy during the testing session according to what we have learnt up to now. In either of these two cases, at the stage of performing the practical test tactic actions, we will be surely confronted with constraints affecting the testing goal's optimal completion.

Let's look more closely at the various categories of testing process constraints. Among the types of constraints impacting the testing mission's optimal completion, most of the types are related to:

1. Time (too short)
2. Cost (too high)
3. Quality (expectations too high – zero defects!)
4. Resources (lack of, insufficient, not trained)
5. Test environment (incomplete, not set up)
6. Test tools (lack of, not suited, difficult to maintain)
7. Test activities (none or wrong testing technique applied)

The first six categories of constraints must be handled by test managers in their continuous risk mitigation activities throughout the testing process life cycle. This is not always an easy task; you must admit that it's hard for a test manager to:

- request some more testing time (testing cost is implied too)
- criticize a manager for a wrongly imposed 'good enough' level of product quality
- ask for an extra tester to help the current testing mission
- ask for some more hardware to set-up the test environment
- convince the company's top management to buy a new testing tool
- instantly become an expert in using a new testing tool

I will focus here only on the last category, the constraints connected to the *test activities*, where often the constraint's root cause is the wrong option for an inappropriate *testing technique*. Here we are fortunately on solid ground, it depends only on us, the testers, to remove these constraints.

The *ISTQB Foundation & Advanced syllabi*[3] describe a complex taxonomy of testing techniques. These are the possible candidates to be used when searching for alternatives in the "evaporating clouds" method. In our previous example on how to remove a constraint, we saw a 'high-level' conflict situation ("execute as many test cases as you can" versus "execute only some of the test cases").

Functional Testing techniques:

1. Specification based (Black box)
   - Equivalence partitioning
   - Boundary values analysis
   - State transition testing
   - Cause-effect graphing
   - Decision tables testing
   - Syntax testing
   - Classification tree method
   - Orthogonal arrays, All pairs
   - Use Case testing
2. Structure based (White box)
   - Statement testing
   - Decision testing
   - Branch testing
   - Condition testing
   - Multiple condition testing
   - Condition determination testing
   - LCSAJ
   - Path testing
3. Defect based
   - Taxonomy
4. Experience based
   - Error guessing
   - Checklist based
   - Exploratory
   - Attacks
5. Static analysis
6. Dynamic analysis

Non-functional Testing types:

1. Accuracy testing
2. Suitability testing
3. Interoperability testing
4. Functional security testing
5. Usability testing
6. Accessibility testing
7. Technical security testing
8. Reliability testing
9. Efficiency testing
10. Maintainability testing
11. Portability testing

Obviously, at the *tactical* 'low-level' in our testing session, we will encounter different kinds of conflict, much more dependent on the current testing context. Here, we must rely on our ability to choose the suitable testing technique in the process of 'evaporating the clouds" of the constraint's conflict. In any such situation, try to draw a quick "needs-wants" graph and then start to identify the wrong assumptions, leading to the "evaporating cloud" solution candidates, selected from the testing techniques you know. Always try to select a "need" that can be satisfied using some other testing technique than the one currently used.

For instance, we may be confronted with the task of testing a dialog which is used to configure a system, and containing combinations of groups of radio-button choices. Here, the constraint may emerge as the conflict between *too many combinations* to be tested and the *limited testing time available*. As possible "solutions" we may use some testing technique candidates:

- *combinatorial testing* techniques (orthogonal arrays, all-pairs testing, t-way testing[6], including the use of tools like AllPairs, PICT, FireEye)
- *cause-effect graphing* and *decision tables*, to identify particular rules that could be hidden behind some particular groups of combinations
- *classification tree method*, to quickly visualize test ideas and find relevant test values

Keep in mind that the testing context (available time, test environment, tester experience, client expectations and feature criticality) will have a serious influence in our choice.

Well, to be honest, our quest does not stop here. There is usually more than one constraint affecting the completion of the testing mission. Our success will be critically influenced by our ability to pick the most important constraint! Here, too, TOC has developed a methodology to help us identify the critical constraint (but, that's another story to be told…please consult the referenced books on TOC[1,2]). And, finally, remember that the testing constraints removal is a continuous process, like the Deming's PDCA cycle. Once we removed a constraint, we will surely find another constraint that affects our testing mission's optimal completion.

Keep TOC's thinking processes as a valuable asset in your first-aid testing kit!

References

1. Eliyahu Goldratt - Theory of Constraints, North River Press
2. William Dettmer - Strategic Navigation: A Systems Approach to Business Strategy, ASQ Press
3. ISTQB - Foundation & Advanced level syllabus
4. Derk-Jan de Grood - TestGoal, Result driven testing, Springer
5. Rex Black – Critical Test Processes, Addison-Wesley
6. Yu Lei & others - IPOG: A General Strategy for T-Way Software Testing, Engineering of Computer-Based Systems, 2007

Biography

Ladislau Szilagyi holds a BS in Computer Science & Mathematics, 1978, Bucharest University, Romania and has more than 30 years of working experience in IT. He worked until 1991 at the Research Institute for Computer Science, Bucharest, authoring multitasking real-time kernels and process control systems. He has been involved in software testing since 1995 and now works as Test Manager at Totalsoft, Romania.

His main specialized areas of consulting and training are Quality Management, Testing and Software Process Improvement. He contributed several articles to the Carnegie Mellon's Software Engineering Institute Repository (https://seir.sei.cmu.edu/seir/), covering topics such as CMMI, Testing, Metrics and Theory of Constraints. He is ISTQB CTFL, member of ISTQB South East European Testing Board and an active ISTQB trainer.

Index Of Advertisers

Erik van Veenendaal is a leading international consultant and trainer, and recognized expert is the area of software testing and quality management. He is the director of Improve Quality Services BV. At EuroStar 1999, 2002 and 2005, he was awarded the best tutorial presentation. In 2007 he received the European Testing Excellence Award for his contribution to the testing profession over the years. He has been working as a test manager and consultant in software quality for almost 20 years.
He has written numerous papers and a number of books, including "The Testing Practitioner", "ISTQB Foundations of Software Testing" and "Testing according to TMap". Erik is also a former part-time senior lecturer at the Eindhoven University of Technology, the vice-president of the International Software Testing Qualifications Board and the vice chair of the TMMi Foundation.

Improving the testing process using the TMMi

by Erik van Veenendaal

The holiday period is always a good time to rethink issues. My column was this time written on the beautiful island of Bonaire in the Dutch Caribean. Facing the ocean and drinking a local beer, I'm trying to focus my mind on test process improvement ……

Different angles
Since the latests developments around the TMMi are a hot topic in the testing community, this is time to dedicate my column around "improving the testing process". Interesting to note that the first **ISTQB** expert level to be released (scheduled for early 2009) is called "Improving the testing process". Many organizations, whether using traditional development models or applying a type of agile development, are trying to improve their testing. To me there are various angles to work from. The one that seems to get most attention is process orientation. Before exploring this more in depth I would like to point out that other approaches such people-oriented ("just make sure your testers are top of the class and they will do a better job") or automation-oriented ("using a unit-test framework and/or automating the regression tests are proven ways to improve the effectiveness and efficiency of testing"). Don't just focus on processes, balance your improvement efforts!

The need for improving the testing process
Many organization use a reference model for test process improvement such as TPI (related to the Dutch testing standard TMap) or TMMi (related to the well-known process improvement model CMMI). Reasons for test process improvement are by now clear to most professional testers:

- Larger and more complex systems are being developed
- IT-systems that play a critical, and sometimes even safety critical, role in the society
- Despite numerous attempts system and software development have not been able to produce low defect density systems
- The amount of time spent on testing in large projects is at least 30%, but often it sums up to 40% or even 50% of the total project effort

In addition we are facing challenges due to short time-to-market requirements, strong competition, new technologies and of course everything around outsourcing. In short improving the testing process is not an option; it is mandatory for those who want to survive and be successful in business. I do not care so much whether one uses TPI or TMMi, both have their strong and weaker areas. More important is the right approach consisting of amongst others a clear mission or policy ("what is it that we are trying to achieve?"), a thorough improvement plan, a practical angle, management commitment and adequate resources.

New developments around the TMMi
Traditionally TPI has been strong in various industries, such a finance. Recently TMMi is gaining "market share" very rapidly. One of the main reasons being its strong relationship in every way to the CMMI. Achieving TMM level 2 almost automatically ensures compliance with the CMMI level 3 process areas Verification en Validation.

The TMMi Foundation has recently been established to coordinate, speed up and professionalize the developments around the TMMi. Most important areas of attention are:

1. Full development and maintenance of the TMMi model. (Currently the TMMi framework and level 2 definition are freely available at www.TMMiFoundation.org.)
2. The development of a formal certification scheme. Like with the CMMI it shall be possible to formally rate according to a predefined procedure the TMMi maturity level of an organization. Of course the requirements for such a scheme are highly related to the experiences with the CMMI.
3. The establishment of requirements for TMMi (lead) assessors; those that formally want to perform TMMi assessments. One needs to have enough knowledge and skills regarding testing, auditing, process improvement and of course TMMi.
4. The TMMi Foundation will in addition be a forum where persons and organizations can exchange experiences and run discussions regarding the TMMi and its usage in the real-life.

Results and uptake

All together a development that will enhance the testing profession. A test department can now achieve a formal, but most of all meaningful, quality mark. I'm aware of many companies that are involved in testoutsourcing that are studying how to become a formal TMMi accredited organization.

I already know organizations that during their road to higher testing maturity have been capable to decrease the test execution lead time and at the same increase their test effectiveness. They also support this claim by having concrete metrics!. Is this utopia? I don't think so with the right approach and attitude it can be done!!

Back to my beer and a refreshing dive in the ocean….

For queries or comments regarding ISTQB expert level, TMMi or the TMMi Foundation you can contact Erik van Veenendaal (eve@improveqs.nl)

A Communit

90,000 Cer

Are you on th

www.is

*July 2008


y Worldwide!
tifications*
e right track?
tqb.org
ISTQB
International Software
Testing Qualifications Board

10 tips for successful testing!

by Dominic Maes and Steven Mertens

Software development is often a balanced exercise in time and quality management. For most companies it is a constant struggle to complete tasks on time and within budget, usually resulting in quality being neglected.
When applications are finally released into production, users are often confronted with only basic functionalities or even the complete omission of functionalities due to the lack of development time.
How do you set up or improve your software testing process to reduce the business risk of poor application implementation?
Here are a few simple tips to get you ahead in quality, saving you time and money:

1. Start early and be prepared!

The key to a successful software development project is preparation. Projects often start as simple ideas with enthusiastic developers committing hours of effort prior to in-depth investigation, this results in unforeseen problems causing delays and project overrun. Testing should be introduced from the outset, this way you have an objective look at the project feasibility and its risks prior to committing valuable effort. In addition the test or quality department should be involved at an early stage to review requirements and design documentation. Reviewing helps to highlight defects and if conducted early in the process will save money by reducing the cost of expensive repairs or corrections. Defects found in the requirements phase are on average 100 times cheaper to correct than the same defects found in production. Preparing your test execution is paramount to getting your project completed in the shortest possible timescale. Valuable time can be saved during the test execution phase by specifying your test scripts and creating a test scenario before the application under test is delivered.

2. Select a testing method which fits your needs

Iterative development, e.g. V-Model, Agile, Extreme Development… Each Software development process needs its own adapted test process; select the testing method correct for your application. You and your management team will benefit from a process that not only reveals the business/project risks but provides the comfort of knowing that the testing team has mitigated these risks. Pay attention to your project specific characteristics, there is no such thing as a 'universal test approach'. Processes that work for one company or project may be totally inappropriate for another.

- Do you need to react quickly and make lots of changes?
- Is the goal of the project to add functionality to an existing application?

Many different aspects of your development process can influence the correct choice of the testing approach. No matter which direction you choose, ensure that your test process is structured and started at an early stage.

3. Be objective

The test team should have an autonomous role within the organization as it is often difficult to work within the project itself. Some project managers tend to neglect the test teams advise when their project deadline is at stake and quality is no longer important. When there is a delay in a development, many project leaders also have a tendency to use the time allocated for test execution to perform further development activities, hence testing is usually skipped and not just postponed. This can have serious consequences for the overall quality of the project which often ends-up in rework when the application is placed in production. The additional work results in delays and further cost to the overall project. When your organization is large enough (i.e. multiple IT projects per year) you can even consider setting up a test factory, a separate department conducting independent and uniform testing of all your existing applications and new projects.

4. Review, review, review, review …

Testing should start early in the development process, you cannot afford to wait for code to start testing your application. Make sure you start reviewing your documents right from the outset; as well reviewed documents tend to result in 65% less defects being carried forward into later phases of the development cycle. This can amount to huge saving of time and money. Don't 'just send' your documents asking colleagues to review them without explanation or preparation, instead give individuals specific assignment on the documents that you want them to review. Have someone check the technical aspects, another, the documents on a higher level and yet another against the layout standards used within the company. This way you can avoid holding review meetings where only grammar mistakes have been highlighted. Reviewing is an exhausting task, so only reviews for short periods, one hour maximum, at a review rate of 10 pages for the period, make sure you are not disturbed. Avoid doing reviews in a crowded or noisy office environment, close your e-mail…. Make sure you become an expert in your reviewing methods. It takes time to learn 'how' to review and code in a professional and organized way.

5. Prioritize

It is generally accepted that testing everything is impossible. In a normal application or system, there are far to many functionalities and to many possibilities to test every single one of them. An application can take several man-years to test so it is important to test those

aspects of your application where the risk of damage or frequency of use is expected to be the highest. A thorough risk assessment should be performed before setting up the test strategy. Select test techniques according to the risks you want to reduce. Build a strategy based on the risks that are involved in the project.
Remember: No Risk, No Test.

6. Measure!

Two important questions which should be posed.

- What do you know about your testing process?
- What do you want to know about your testing process?

Most of the time, it's easy to establish the cost to solve a defect in your application (at different stages), but what is the cost and how efficient is your team in finding these defects?
The key to answering these questions is: - start measuring.
Gather data to monitor your test process, efficiency and effectiveness of your test department.
Utilize metrics which provide the information you need to know about your process, i.e. test scenario density, defect density, number of defects found in the first month of production, etc…
The question is often asked 'When is the best time to start measuring?'
In our experience, NOW is a good time to establish metrics, the sooner you start collecting data the faster you can start improving.

7. Know when to stop testing.

When can you suspend test execution?
As a rule, you should stop 'Bug Hunting' when the cost of finding an additional defect is no longer in balance with the risk you're exposed to. Under normal circumstances, you need to set exit criteria to decide in advance when testing should stop, for example, define a criteria that will stop testing when 90% of the available test scripts has been executed with a maximum of 5 open defects. Make sure your project criteria are clearly defined. Discuss your exit criteria at the start of the test project and incorporate them within your test plan, this will avoid discussions at a later date. When no exit criteria are defined, testers tend to continue testing until all defects have been found or solved.

8. Preserve.

The strength of software testing partially lies in the structured preservation of testware. On completion of testing, take some time to evaluate and assess your deliverables and preserve them for future use. Test cases should be stored in such a way that they can be reused in later projects- or even during the maintenance phase. You should always have your 'old' test cases ready for immediate re-use, this way you can save 30% to 50% in overall test effort.

9. Communicate

In many companies there is an 'It is not MY responsibility' attitude. Communication is poor between the end-users, architects, developers, testers and management. Too often this kind of behavior leads to misinterpretation in requirements, unwanted functionalities and unsatisfied end-users. Communication is key to 'quality'.
Listen to the end-users and get involvement of the complete team to obtain the best results.
Be diplomatic and have constructive discussions.
Software development is constantly balancing budget, quality and scope.

10. Last but not least: keep it simple!

Often companies become lost in administrative procedures when considering testing. An efficient test process should excel in its simplicity, no mountains of red tape, nor tedious ways of giving your quality team and developers a hard time. Choose an approach that fits your project or company and does the job: assuring quality for your applications.

Good luck!

Biography

Dominic is a senior test consultant at TOBIUS NV.
Dominic has a master degree in commercial and financial sciences. Since 1997 Dominic is working as a professional test consultant. He started from junior tester over test coordinator and test manager to an expert level, specialized in both test processes and tools. Dominic gained experience various industries and sectors, such as retail, telecom, manufacturing, finance and government in Belgium and the Netherlands. Dominic has excellent knowledge of TMap® and TMap Next®, and is both ISEB Practitioner and TMap® Certified. He is a requested speaker at colleges, universities and seminars.

Steven is Manager of the Competence Center Testing at TOBIUS NV. Steven has a bachelor in management and insurances. Since 1998 Steven gained experience in various industries such as the pharmaceutical industry, gas & electricity, social security, insurance, banking, retail, telecom, industry and government in Belgium, the Netherlands and Luxemburg. He has an excellent knowledge of testing methodologies such as TMap®, STBox™ and Test Process Improvement frameworks. Steven is one of the three creators of STBox™. He has experience in coaching people, giving training, test management and team leadership, determining test strategies, writing test plans and practical implementation of test methodologies. Steven is ISEB Practitioner Certified since 2005. He is a frequent speaker at colleges, universities and seminars.

Explaining Security Testing as Real Trousers

by Bogdan Bereza-Jarociński

Motto:
"I reckon our bard wasn't expecting flames to shoot out of the floor unexpectedly," said Cohen.
Truckle shrugged theatrically.
"Well, if you're not going to expect unexpected flames, what's the point of going anywhere?"
(Terry Pratchett "The Last Hero")

Abstract

"Can you perform security testing for us? ", asked a prospective customer.
"Sure, we are test specialists. Give us your security requirements, and we will test them! ", was the answer.
In reality, the situation is not that simple. In this respect, security is like usability: you can seldom expect customers to know all about it and get a free ride on their requirements. You may have to provide your customer with information about what "security" actually is and how it should be tested.
Besides, whereas security requirements may be simple ("no unauthorised access shall be possible"), testing for compliance with them may not be. Security testing is very much looking for unexpected side-effects were noone (except hackers or crackers) expects them, which requires minute technical knowledge and plenty of "error-guessing". For some reason, all this is called "penetration testing" in a security context.

What Is "Security"?

There was once a small company making clothes. One day they got an interesting proposal: to produce jackets and trousers for mountaineers. One clause in this proposal required that "you must prove that your products fulfil our safety and security requirements". However, no such requirements were explicitly stated. After some telephone conversations with the wholesaler who placed this enquiry, it was clear that no description of security requirements existed - it was up to the producer to convince the wholesaler that the product was both "safe" and "secure" - and to create the definition of security on the fly.
Some chaotic discussion at the clothes maker's boardroom followed.
- Secure trousers must not endanger mountaineers' safety. This means perhaps that they have to have two legs, not just one leg, and be waterproof enough to protect their users from rain and snow - said the Chairman.
- Sorry, but I think two trouser legs is more like trousers' functionality, not their "security", whatever it means. Security would be rather that these legs are long enough to go all the way down to the boots, and wide enough to have room for thick fleece trousers under them - said the Vice-chairman.
It must have been a democratic company if anyone ventured to challenge the Chairman's opinion, which was good - otherwise this story could not have been created at all.
- "I guess you have a point here", the Chairman agreed.
- However, what you tell is functionality, too. "Sure enough, trousers for mountaineers must cover them well enough to give protection against the elements, and have room enough to be used over all other clothing, but this does not mean they are necessarily secure. Secure trousers must be reliable: they must not break and fall apart, for example, or dissolve in cold water, or have parts you can stumble on etc.", continued the Chairman.
- "OK, folks, we cannot solve this now. Let's have a meeting in the afternoon - is 3 o'clock OK for everybody? - and let's have some technical people with us", concluded the Chairman.

Security Definitions

Here are some notes that were found by the cleaners on the boardroom floor in the morning of the next day – they were the result of a prolonged brainstorming activity.
"Colour matters, but not for security - or should it be well visible, like signal red for example?"
"Safety = trousers do not malfunction in a way which puts their user in danger. If they for example fall apart while stored in a storeroom which is too damp, this is a problem, but not a safety problem".
Some caps from beer bottles were found in the dustbin. They were the result of a discussion on the difference between mountaineers' "safety" and "security", which continued until the wee hours and was terminated only when the amount of consumed beer seriously hampered the participants' ability for deep philosophical reasoning.
"Our problem is, we have to prove that trousers are secure". The word "prove" was crossed out and "test" written above it; someone pencilled "verify" beside the whole sentence.
"We have to know what secure means, then we have to prove that it is".
"You cannot prove it unless you actually use them"- note was still glued to another note, stating that "too expensive to prove first when all is ready for mass production, we must prove it from the very beginning, otherwise wrong fabric and thread may be ordered!".
An officially looking document lay on the table beside the Chairman's leather armchair. It was mostly empty, but contained one phrase printed in fat style: "What is meant by security is defined by the way we have verified it, which is described in Appendix 17".

Where to Look for Security (and Safety) Bugs

Rumours about the trouser issue started to spread across all levels of the company. Employees began to discuss it. An overheard conversation was recorded:

- "You know, Frank, this trouser security stuff reminds me of when I once tried to install some extra electrical appliance in my car and thought I could save some money by doing it myself. I had to connect some electrical wires and it worked perfectly! However, you got an electric shock in your fingers when you touched anything in the car afterwards. I kind of did too much: the stuff worked all right, but it did something more, too. And what it did was definitely not safe for you!"
- "So you mean, for these trousers it could be something like we make them good, but some idiot adds a build-in parachute which opens when wind is strong enough. And then someone gets injured because this parachute opens suddenly in strong wind during climbing. You do too much, and it may result in things becoming less secure than when you do too little."
- "Absolutely. This security stuff will be a bit like finding out where you can expect unexpected things to happen. And, let me confess, I even checked the difference in the dictionary, and it looks to me now like some things may be unsafe when insecure, and some may be insecure when unsafe!"

The Company's network administrator who had worked for an Internet company before it went bust two years ago.

- "Security without network connection is easy", he said.
- "You cannot easily compare trousers to computer systems, because they are not connected to anything or to one another. If they were, anyone could at any time try to make your climber's trousers disappear, or put patches on them where there should be seams, or simply get into the same pair of trousers while someone uses them already. This is about what we call "viruses" and "hacking" in our computer world. With trousers, it will be easy: no connectivity, no software configurations… really easy!"

Security Testing

Because security seemed to be very important to the wholesaler, the manager responsible for production and quality assurance decided to assign a separate team to this job. "The first thing", he thought, "is that you have to ensure that the trousers do not create any dangers specific for mountaineering. That means you need someone who knows about mountaineering, otherwise it would all be based on blind guesswork. On the other hand, it is not enough to ask a mountaineer, who knows much about what he or she does in the mountains, but perhaps little on how trousers are made. A mountaineer can make the trousers break - good if it happens before mass production has started! - but he cannot tell us why it has happened: Was the fabric too weak, or the thread, or the seams too thin, or does perhaps the sewing machine's needle weaken the fabric?"

- "A mountaineer needs to guess what can in reality happen in the mountains, and my technical person would need to guess what can go wrong from the technical perspective. When something breaks, it will be necessary to determine why - again, technical experience would be necessary for that", thought the production manager.
- "And the difficult part", continued the thought process in the production manager's harassed brain, "is that testing in the mountains requires whole trousers as well as real or at least realistic imitations of mountain environment. My technical person will have to follow my mountaineer everywhere and keep an eye on what happens with the trousers. Difficult, that one! "

How Much to Test Security?

- "Never!" shouted the Managing Director and rammed his fist into the shiny desktop.
- "Never, never, never!" he repeated and shook the offensive document violently.

It was the trouser development project's budget proposal.

- "Two bloody million just to be able to tell the -ing wholesaler that our trousers are secure? " came the next broadside.
- "Let him put his ass into those trousers himself and let him go to the mountains, he would be able to see for himself if they are good enough or not!"
- "The problem is", retorted the technical manager, "that the wholesaler wants to be convinced that our trousers are secure without having to make a mountain trip himself. Besides, he says that for his customers, his own mountain-walking habits may be not enough. Many customers are half his age and three times his physical temperament, and they have to be convinced that our trousers will serve them even in extreme conditions. "

There was silence, an ill-omened silence, and finally Managing Director asked very, very quietly:

- "OK then, if it is so important, why spend just two million? Why not ten million? If we did that, we would be still surer that we really, really know how secure our trousers are?"

But sarcasm did not bite. The answer was prompt and very much to the point.

- "Well, this is exactly how we reasoned while making this budget. We got three mountaineers here on a short-contract basis and they made for us what we call a "usage profile": how we guess these trousers will actually be used. Then we made an estimation of how much security is needed. You can make perfectly secure trousers, but they would cost a huge amount of money to produce. So our estimation was based on how much extra we guessed customers would be willing to pay for trousers with our "security certificate" logotype. And how much these trousers would be allowed to fail under extremely extreme conditions without compromising the value of the certificate. Finally, we tried to figure out how and how much to test in order to achieve that level of quality with sufficient confidence. And two million is what we got. "

More silence, but not so much ill-omened as before.

- "We do have a lower proposal",, continued the technical manager, "complete security testing for only 700 thousand, but I really believe the risk would be too high. And a really expensive one, for three million, but that would mean risking that the trousers are too expensive or unable to be sold, or our profit margins dangerously low."

Really impressive, the way the technical manager convinced the Managing Director. He was well prepared for this meeting and produced convincing, supporting numbers with his reasoning. Just like we are used to in SW testing in general, aren't we?

Dragon's Voonerables

"What went wrong before", said some pessimistic employee at our clothing manufacturer, "may go wrong in future, too. "

Some impressionable manager who happened to be within earshot thought it was a great idea and suggested creating a list of all remembered past security-related failures, then making a checklist out of it. Here are some excerpts from this checklist.

Implementation Faults

A consignment of party dresses was once sewn with seams so flimsy that some dresses simply fell to pieces when partying was wild enough. In some cases users considered this almost a required functionality, but in some cases a security (and safety) hazard.

Metal zips were once used in winter jackets. There were complaints that they sometimes froze and could not be opened. Just a minor flaw for ordinary clothes, perhaps a security (and safety) issue for mountaineering clothes. It was decided that only plastic zips will be used, and that they will have to be tested for durability in low temperatures.

Insecure Design

A similar failure list was created for design-related failures. Generally, it was agreed that the shape of trousers must give maximum mobility and must not create any points of high tension and risk of fabric splitting. Besides, all details such as zips, buttons and clips must be designed so that they can be operated even with thick gloves. Ahem... was it functionality, perhaps?

Unexpected input, shared code libraries, Internet applications

Putting a hot gasoline stove instead of your leg into a trouser leg could definitely be classified as an "unexpected input" and as quite a major hazard, but it was decided as unavoidable.

On the other hand, other "unexpected inputs", such as different kinds of fabric used for undergarments coming into contact with trousers' fabric was considered a legitimate safety issue.

Decision was unanimously taken to have a closer look at other forms of unusual usage with possible safety and security consequences, such as what happens when the trousers are left for several days in a frozen tent.

Usability

Usability, defined as "ease of use" and "suitability for intended usage", is indeed important for mountaineers' clothes! But who knows what matters for usability? Definitely not the clothing manufacturer, and not the wholesaler. It is the end-user's opinion that matters.

Here, the mountaineers who belonged to the security-testing team produced a long list of what was important. This list was immediately given to those responsible for the trouser design so that - hopefully - the product would fulfil all usability requirements from the beginning, thus eliminating the need for expensive re-design and re-configuration of all equipment on the trouser production line. Here are some excerpts from this list.

- You must be able to put on and take off your trousers with thick gloves on your hands, in complete darkness, in confined space (tent or bivouac), or in strong wind.
- Trousers must be equipped with a loop for securing, so that they do not fall down or get blown away. "Is that usability or function, by the way? ", thought one mountaineer while preparing this list.
- Otherwise, there should be no loose elements outside which could catch in other equipment or rocks.
- A flap enabling you to easily answer a "call of nature" must be provided. It must never open accidentally, and must never freeze, either! By the way, function or characteristics - again?
- All buttons, zips and clips must be made of plastic so that they are not cold and never freeze. This plastic must not become brittle even after long exposure to very low temperatures.
- Pockets (a long treatise on pockets followed, omitted here).

Load and performance

Mountaineers clothes must be robust and withstand quite brutal usage. Tear and wear must be gradual, so that weaknesses are easily visible and security degradation predictable. Never must mountaineering equipment break suddenly.

From the customer's (and even seller's) point of view, it is of course important that garments are long-lived, withstand storage conditions well etc, but these aspects of their performance are not important for security.

Mountaineers use their clothes in extreme conditions, which must be taken into account when measuring their performance. Damp conditions, wind, water and temperatures well below zero are common.

On the other hand, perfectly water-proof and wind-proof trousers may be too heavy and stiff, and their "breathing" ability may become poorer, too. A proper balance must be defined and achieved in this respect.

Good news is, the fabric's, thread's, button's and clip's strength, resistance to friction, water etc. are readily measurable and some data on what is appropriate for mountaineering clothes is available. The important thing was not to forget to test the resistance of the whole trousers, and not just their components. Direction of typical pressure, points where extra pressure is applied (like knees) must be identified: in other words, you need an "operational profile" for trouser users, just like for Internet users!

Organisational aspects

From the beginning, it was thought that the way the trousers should be used need not be explained to anybody, or at least be left to the users themselves. However, finally it was agreed that even the manufacturer had some responsibility for telling customers how to use trousers securely: by providing appropriate cleaning and storage instructions, for example, and a sufficiently detailed description of how much load and performance to expect from them.

Security Testing Process

Armed with all definitions, checklists, tools for exercising fabric's resistance and other weaponry from the quality assurance arsenal, the manufacturer's personnel stood braced for the challenge of entering the realm of production of trousers for mountaineers. And they succeeded - no doubt (?) due to their meticulous preparations and thought processes described above. There are quite a few manufacturers of good, robust, secure trousers for mountaineers, so you can easily visit the nearest shop with outdoor equipment and study the result.

Below follows the description of how they did it. Nobody would disclose the well-guarded industrial secret whether this process description preceded or followed its actual first-time execution, but here it is, and the manufacturer is now well-prepared for more challenges of the same kind.

Requirements Engineering

The production manager went hiking to the mountains. A famous mountaineer was invited to the factory and participated in the process, bribed by free trousers for his next expedition to the Himalayas (money for that was taken from the company's marketing budget). Trousers already available on this market were bought and analysed in detail.

A list of requirements was produced. This list was not perfect, but could be completed later on, as forgotten requirements were discovered, some of them as late as during monitoring of customers' complaints.

Design

As described in section [Insecure Design], the design of trousers was carefully verified. Besides, all rules used in the design process were re-used later on as test cases to check whether the final product fulfilled them.

One pair of trousers was quickly made and worn - by the mountaineer, followed by the panting production manager - during a one-day hike in the Ben Nevis region. Some obvious design flaws could be identified and removed immediately afterwards.

Component Testing

All components were carefully checked for compliance with known requirements. Many requirements were not security-related (price, availability, pattern, suitability for production process etc.). Some were security-related, such as tear resistance, resistance to wind and water, characteristics in prolonged damp conditions and in temperatures below zero.

Integration Testing

After parts of garment were sewn together, they were put to tests checking their resistance, shape etc. Some faults were discovered and could be removed at this stage. The type of thread used for main the seams happened to be too thick and impaired the fabric's tear resistance. One type of needle used in sewing machines made holes, which were difficult to keep waterproof afterwards.

System Testing

A few first pairs of trousers left the production line! They were minutely inspected for manufacturing faults, extensively tested for all kinds of performance, checked for accordance with initial design. Acceptance testing was started in parallel: the next visit to Ben Nevis region took place.

High-level Integration Testing

F… (they really meant "heck")! Suspenders' clips tended to fasten in jacket's inner pocket zip. The distance between jacket's lower hem and trousers' upper hem was too little, snow could get in. The trouser legs were too narrow for those huge plastic boots used for crampons. The side-pocket zip was sharp and could make holes in down sleeping bags, should someone choose to sleep with trousers on.

If anyone suspects that some of these faults were actually discovered during acceptance test hikes at Ben Nevis, I must confess this. They were still, however, typical high-level integration faults!

Acceptance Testing

While the contract mountaineer was busy acceptance-testing-mountaineering, some of his colleagues got free trousers for their winter climbs in the Alps (beta-testing). Young personnel from production and marketing departments got some free trousers, too, for use while walking, hiking or jogging (alpha-testing).

Oops! A female climber discovered some problems with the location of the "call-of-nature"-flap. Instead of complete re-design, a separate model for women was introduced.

Monitoring in Operation

Buyers were given a "registration card" to fill in (voluntarily). If they answered some questions, their "registration cards" participated in a lottery, where mountain tents were the main prizes.

These registration cards, as well as of complaints from users later on, were analysed - among other aspects - for security-related issues.

Books

Besnard, J., F. Developing Software for Safety Critical Systems
Cole, E. Hackers Beware: Defending Your Network From The Wiley Hacker
Friedman, M., A., Voas, J., M. Software Assessment: Reliability, Safety, Testability
Gardiner, S. (ed) Testing Safety-Related Software: A Practical Handbook
Ghosh, A., K. E-Commerce Security: Weak Links, Best Defenses
Hatton, L. Safer C: Developing Software for High-Integrity and Safety-Critical Systems
Herrmann, D., S. Software Safety and Reliability: Techniques, Approaches, and Standards of Key Industrial Sectors
Jamsa, K. Hacker Proof: The Ultimate Guide to Network Security
Leveson, N. Safeware: System Safety and Computers
McClure, S., Scambray, J., Kurtz, G. Hacking Exposed
McGraw, G., Felten, E. Securing Java: Getting Down to Business with Mobile Code
McGraw, G., Viega, J. Building Secure Software
Pipkin D., L. Halting the Hacker: A Practical Guide to Computer Security
Redmill, F. System Safety: Hazop and Software Hazop
Schiffman, M. Hacker's Challenge: Test Your Incident Response Skills Using 20 Scenarios
Sindell, K. Safety Net: Protecting Your Business on the Internet
Smith D., J. Achieving Quality Software - Including its Application to Safety-related Systems
Van Der Meulen, M. Definitions for Hardware and Software Safety Engineers
Villemeur, A. Reliability, Availability, Maintainability and Safety Assessment
Whittacker, J. How to break software
Whittacker, J., Thompson, H. How to break SW Security
Wiegers K., E. Software Requirements

Organisations, companies, services and standards

Board of Certified Safety Professionals, www.bcsp.org
BS 7799, www.bsi.org.uk
Bugtraq, mailing list at www.securityfocus.com
Cigital, www.cigital.com (books, training and white papers on security)
COAST Software Vulnerabilities Testing Group, www.cerias.purdue.edu/coast/projects/vuln_test.html
Common Criteria, http://csrc.nist.gov/cc ("Common Criteria for IT Security Evaluation, Common Language to Express Common Needs ")
IEEE Standard for Software Safety Plans IEEE Std 1228-1994 (www.ieee.org)
Internet Security Systems, www.iss.net
LogiGear, www.logigear.com (Internet security testing, training)
NASA-STD-8719.13A (SW Safety Standard), www.hq.nasa.gov
Packetstorm Security, http://packetstorm.linuxsecurity.com
"Penetration Testing": www.google.com gives 128,000 results
Safeware Engineering Corporation, www.safeware-eng.com ("applied research and development for real-time, safety-critical systems")
Security Focus, www.securityfocus.com
Security Innovation, www.sisecure.com (security: tools, training, testing)
www.hackers.com
www.hackersoftwarehacker.net
www.insecure.org

Dedicated Tools

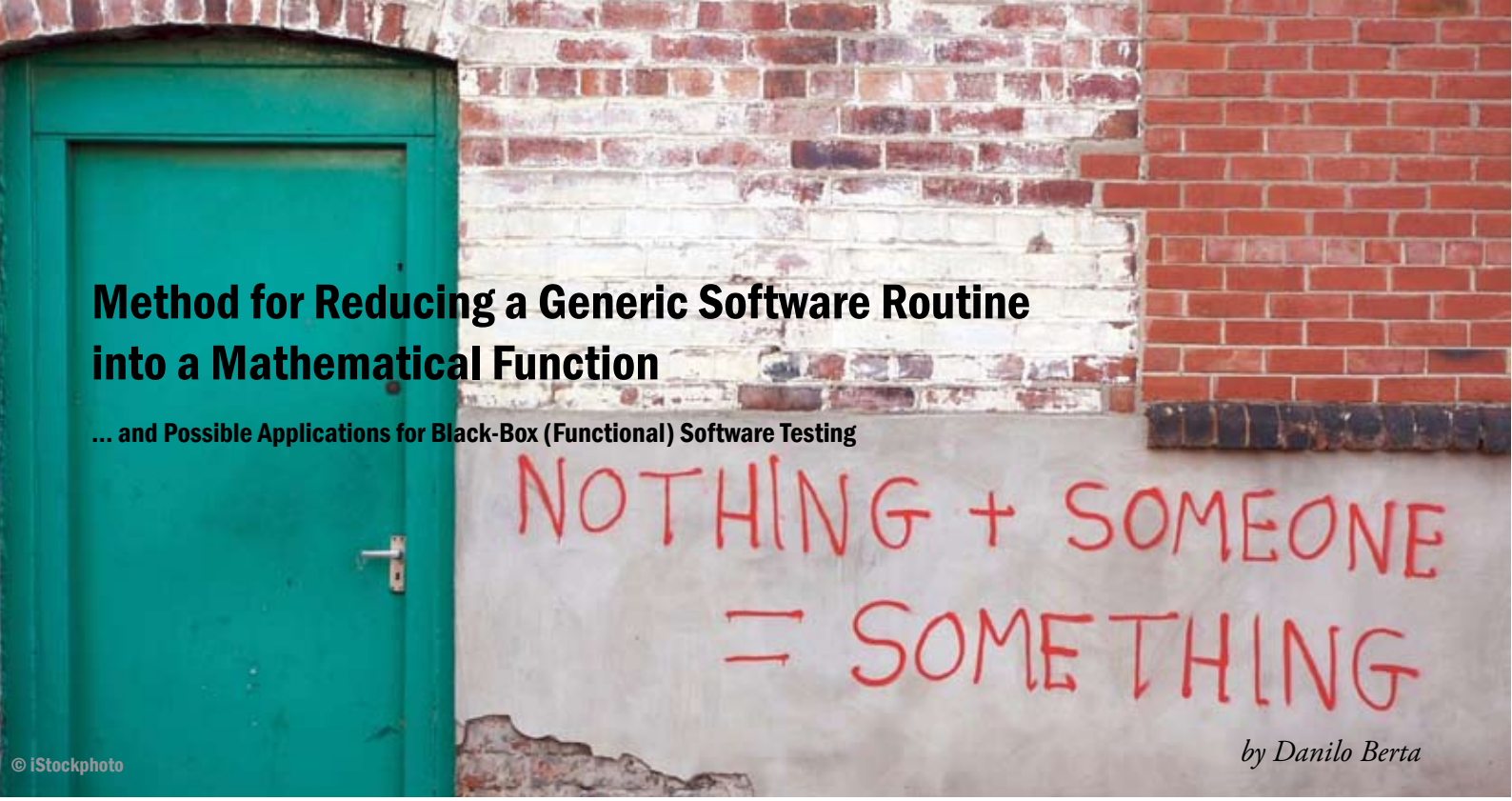
GFiLANguard, www.all-internet-security.com/free_protocol_analyzer.html
Holodeck, www.sisecure.com/products.html
Internet Security Systems, www.iss.net
Nessus, www.nessus.org (free security scanner)
SAINT, www.saintcorporation.com (vulnerability scanner)
SATAN (Security Analysis Tool for Analysing Networks), www.fish.com/satan
www.insecure.org/tools.html

Biography

Bogdan has experience in teaching testing, requirements and other SW engineering subjects since 1994 in Sweden, Poland, Denmark, Finland, Spain, Luxembourg and the Czech Republic. He pioneered the ISEB Foundation qualification in Sweden (1999), in Poland (2004), then ISTQB® Foundation in Poland (in both English and Polish) in 2006-2007, and now in Russia.

Besides that, Bogdan has been involved in work with standards and syllabi.

He holds both the ISEB Practitioner and full ISTQB Advanced Certificates.
Today, he cooperates closely with the WCSQ 2008 organization.



Method for Reducing a Generic Software Routine into a Mathematical Function

... and Possible Applications for Black-Box (Functional) Software Testing

by Danilo Berta

1. Abstracts

This article describes a method that can be used to reduce a general software routine, which accepts input of alphanumeric values and returns alphanumeric output values, into a mathematical function. A generic software routine takes and produces alphanumeric strings as inputs/outputs, whereby there is no way to define an order in these strings (e.g. it is not possible to determine between two strings which of them comes before or after the other).

In order to explain the matter in more detail, we will consider a simple software routine which produces as output the reverse string that it has taken as input. This routine can be defined as a mathematical function in this way $f(x_1x_2...x_n)=x_nx_{n-1}...x_1$, where $x_nx_{n-1}...x_1$ are generic alphanumeric characters. If we want to draw a Cartesian graph of the function, we need to know if, for example, the string $x_nx_{n-1}...x_1$ comes before or after the other string $x_nx_1...x_{n-5}$, or any other string which we can use as input of the function.

If we were able to reduce a software routine into a well-defined function, we would also be able to use the well-known mathematical methods to test the same function, sampling some of the input values of the function (in some statistical way) in order to verify whether the outputs of the same function for these input values are as expected. This method, which is strictly speaking a black-box method, can also be applied to any routine with any number of inputs and outputs; it's clear that in these cases we have to deal with multi-variable functions. In this article, we will not deal with statistical sampling methods, but will only describe a method for associating a generic string with a number in an unambiguous way, and will then discuss how to apply this method for the test and the regression test of generic software routines.

2. Introduction with a simple example

Let's start with a very simple example to so we can follow the course of the subsequent discussion with a more technical and mathematical approach.

To keep things simple, we assume that we have a really straightforward simple software routine that produces as output exactly the string it takes as input, whatever the string may be. This function will be:

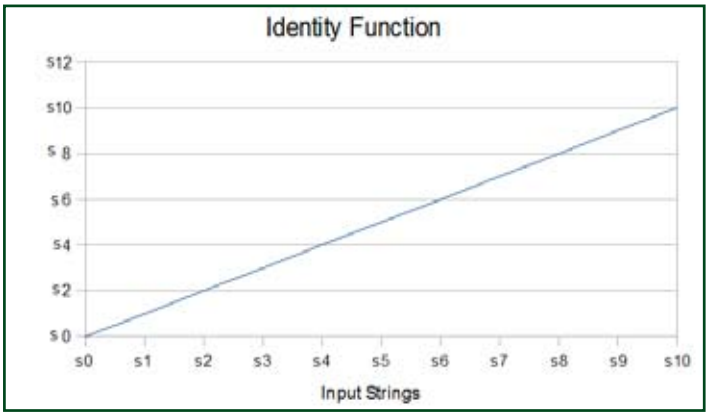


Fig 1: Identity Function Graph Example

(2.1) $f(x_1x_2...x_n)=x_nx_{n-1}...x_1$ where $x_1x_2...x_n$ are generic alphanumeric characters.

If we wanted to draw a Cartesian graph of this function, the first problem we would encounter is how to allocate an order to the input and output string, to be able to define the metrics (the intervals) to use for the X and Y-axes. Intuitively, we can imagine that this function would be the diagonal that splits the Ist quadrant in 2 parts of 45°, as shown in the picture below:

The intervals are labeled as s1, s2,...,s10, meaning that each point represents an input string and s(n)< s(n+1). However, we don't know, for example, whether the input string "ASER" is less than (or greater than) the string "ATER".

We need to define a method to associate a unique number to a string and a method to deduce the string that corresponds to that number.

The first thing that comes to mind is to number the single string elements progressively. If, for example, we suppose that the single elements that compose our input/output strings shall be the first three letters of the alphabet, 'A', 'B' and 'C', we can compile the following table:

Number	String	Number	String	Number	String	Number	String
1	A	13	AAA	25	BBA	37	CCA
2	B	14	AAB	26	BBB	38	CCB
3	C	15	AAC	27	BBC	39	CCC
4	AA	16	ABA	28	BCA	40	AAAA
5	AB	17	ABB	29	BCB	41	AAAB
6	AC	18	ABC	30	BCC	42	AAAC
7	BA	19	ACA	31	CAA	43	AABA
8	BB	20	ACB	32	CAB	44	AABB
9	BC	21	ACC	33	CAC	45	AABC
10	CA	22	BAA	34	CBA	46	AACA
11	CB	23	BAB	35	CBB	47	AACB
12	CC	24	BAC	36	CBC	48	...

Table 1: Association of string with numbers

The table should be read in this way: number 1 is linked to string "A", number 2 is linked to string "B",..., number 46 is linked to string "AACA" and so on.
Clearly, it's possible to go on indefinitely to compile the table above. We need to define a rule to calculate the corresponding number from the string.

The rule is as follows:

Start with the base association of string with number:

Number	String
1	A
2	B
3	C

Table 2: Association of Base String with numbers

As a hypothesis, we will suppose that all strings considered must come from any combination of the base letters {A,B,C} which in the following we will call the **dictionary**. So, string "ACABCCAACC" is valid, while string "ASDFGHAAABBCRR" is not valid, because it contains the elements S,D,F,G,R that are not in the defined dictionary.

The rule is expressed in the following formula, which for now we will describe by example (and we will explain later in detail).

(2.2)AABC = 1*33 + 1*32 + 2*31 + 3*30 = 1*27 + 1*9 + 2*3 +3*1 = 45

This is exactly the number that we find for string "AABC" in Table 1 above. If we examine the method of obtaining a value for string "AABC" it's evident that it's very similar to the way that numbers are usually transformed into base 2, 8 or 16 or any other base into decimal. So, there's nothing new or magic about it. The number "3" we have used in our formula is simply the base of the dictionary, or, in other words, the number of base elements in the dictionary {A,B,C} is exactly equal to three.

This method can be extended for dictionaries with any number of elements. The higher the number of elements in the dictionary, the greater will be the number we deduce for "short strings" (whereby "short strings" means strings with a low number of base elements).

It's a little trickier to find the "opposite rule", i.e. the rule to transform a generic number into a string composed only of elements of some well defined dictionary. For now, we introduce the rule and apply it with an example:

1. Divide the number for the base of the dictionary, considering both the **result** of the division and the **remainder** of the division. For example, for dictionary {A,B,C} the base is 3, as previously stated.
2. If the remainder is zero, subtract 1 from the result and replace the remainder number with the base.
3. If the result is greater than or equal to the base, continue with the division, applying rules (1) and (2) until the result is zero. In our example, this will happen just after we have found a result of the division that is less than 3 and after exactly 4 iterations.
4. Write down the string composed by the remainder of the division, starting from the last, and written sequentially.
5. Replace the number in the string, which was derived as described in point (4), with the letter in the base association string to number of the dictionary (see Table 2).

To explain this in more detail, we will use the number 45 as an example to verify whether the rule is correct. We will expect to find string "AABC", using the dictionary {A,B,C}.

Here is an example of the algorithm's application:

1. 45/3 = 15 remainder 0. Subtract 1 from 15: 15-1 = 14 that is greater than 3, so we have to continue. Replace the remainder 0 with 3.
2. 14/3 = 4 remainder 2. No need to subtract. 4 is greater than 3, continue.
3. 4/3 = 1 remainder 1. The result is less than 3. One division more and then we will stop.
4. 1/3 = 0 remainder 1. Stop here.

So, starting from the last remainder (step #4) and continuing with the other remainders , we have found the numeric string "1123", and using the table in Fig. 2.3 we can now replace the numbers with letters as follows:

$$1 \rightarrow A$$
$$2 \rightarrow B$$
$$3 \rightarrow C$$

In this way, the numeric string "1123" becomes "AABC", which is the string corresponding to the number 45 in the Table 1.

In the following paragraph, mathematical proof of these rules is presented.

3. Mathematical proof of the rules

First of all, we will start with some useful definitions, as used before:

1. A **dictionary** is a collection of **elements** a1,a2,...,an composing the string. It's written as follows: D{a1,a2,...,an}.
2. A **base** of the dictionary D is a number equal to the number of elements of the dictionary.
3. A string is **valid** with regard to dictionary D if it's composed exclusively by elements of D. It's **not valid** if there is at least one element that does not belong to dictionary D.
4. The **length** of a string is a number equal to the number of elements the string is composed of.
5. The **numbers equivalent to the dictionary's elements** (in the following NEDE) are defined by a matrix as follows:

NEDE	Element
$\alpha_1 = 1$	a_1
$\alpha_2 = 2$	a_2
$\alpha_3 = 3$	a_3
...	...
$\alpha_p = p$	a_p

Table 3: Association of Base String with numbers

It is presumed that all α where j=(1,2,...,L) are not zero. This is a fundamental hypothesis for what follows.
Please, note that all α_j for all j=(1,2,...,L) are strictly less than base B of the dictionary.
The first role does not have to be proven, but must be taken as a definition of the algorithm used to associate a number to a string. So, we can state the following:
Definition #1: Given a dictionary D, it's possible to associate an unique number N_0 to any valid string a1a2a3,…aL for this dictionary by applying the following rule:

(3.1) $N_0 =$

$$a_1 B^{L-1} + a_2 B^{L-2} + \ldots + a_L B^0 =$$

$$a_1 B^{L-1} + a_2 B^{L-2} + \ldots + a_L \quad \text{because } B^0 = 1$$

Where B is the dictionary's base and α_j is the numerical equivalent to the generic elements α_j of the dictionary, where j=(1,2,...,L) and L is the length of the string.
Starting from this definition, it's possible to deduce the rule for transforming a generic number into a string composed only by elements of the dictionary.

If we start dividing the number N_0 by base B, we will obtain:

(3.2)

$$\frac{N_0}{B} = a_1 B^{L-2} + a_2 B^{L-3} + ... + a_{L-1} + \frac{a_L}{B}$$

With the position:

(3.3)

$$N_1 = a_1 B^{L-2} + a_2 B^{L-3} + ... + a_{L-1}$$

We have:

(3.4) $$\frac{N_0}{B} = N_1 + \frac{a_L}{B}$$

that is equivalent to $N_0 = N_1 B + a_L$

Continuing in this way, we obtain:

$$N_0 = N_1 B + a_L$$
$$N_1 = N_2 B + a_{L-1}$$
$$N_2 = N_3 B + a_{L-2}$$

(3.5) .....

$$N_{L-2} = N_{L-1} B + a_2$$
$$N_{L-1} = a_1$$
$$N_L = 0$$

From the last equation in (3.5), i.e. $N_{L-1} = \alpha_1$ it should be evident that if we start from equation (3.1), we divide N_0 by $B^{(L-2)}$. For a better understanding, an example is given below where we apply (3.5) for the case where L = 4.

Start with: $N_0 = a_1 B^3 + a_2 B^2 + a_3 B + a_4$

Put: $N_1 = a_1 B^2 + a_2 B + a_3$

we have: $N_0 = N_1 B + a_4$

(3.5 ex)

Put: $N_2 = a_1 B + a_2$

we have: $N_1 = N_2 B + a_3$

Put: $N_3 = a_1$

we have: $N_2 = N_3 B + a_2$

Put: $N_4 = 0$

we have: $N_3 = N_4 B + a_1 \equiv a_1$

In the example L = 4, (L-1) = 3. The next N4 value is exactly equal to 0 (zero), because we divide N_3 (that is less than B) by B, obtaining 0 with remainder $N_3 = \alpha_1$. We have to stop the process at this point.
Remember that we have supposed that all α_j where j=(1,2,...,L) are not zero.

In formula (3.5), the α_j where j=(1,2,...,L) are the remainder of the subsequent division of number N_0 for the base of the dictionary, while the N_j where j=(1,2,...,L) are the results of the division.

From the above discussion we can deduce the following rule to extract from a number N_0 a string of length L composed only of elements from a given dictionary D{a_1,a_2,...,a_n}.

1. Define the dictionary and the number-equivalent elements.
2. Divide number N_0 for the base B of the dictionary, considering both the **result** of the division and the **remainder** of the division, and continue in this way until the result of the division is zero. This happens after L steps.
3. Starting from the last remainder create a string composed by the remainders of the divisions.
4. Consider the remainders of the divisions as the numbers equivalent to the dictionary's elements, and replace these numbers with the equivalent elements of the dictionary, using the "NEDE" table.

Now, all this works fine as long as we find only remainders of the divisions that are non zero. However, what happens if the remainders of the division are zero?

We will use the following simple tip. If the result of the division of X through Y is exactly N, with remainder 0 (zero), it can also be written as follows:

(3.6)

$$\frac{N}{Y} = N \text{ remainder } 0 \rightarrow \frac{X}{Y} = (N-1) \text{ remainder Y}$$

In other words, the result of $\frac{X}{Y}$ should be also be considered as equal to N-1 with remainder Y. For example: 15/5 = 3 remainder 0, but it's also 15/5 = 2 with remainder 5.

So, if we found some α_j equal to zero in the division process in (3.5), we now apply the tip and replace the result with (result -1) and the remainder with the base B of the dictionary.

This results in the following formula:

(3.7)

$N_{L-r} = N_{L-r} B + a_r$ with $a_r = 0$ means: $N_{L-r} = N_{L-r} B$

In other words:

(3.8) $$N_{L-r} = N_{L-r} B = (N_{L-r} - 1)B + B$$

This formula demonstrates the tip explained above.

4. Summary of the rules

In this paragraph, we will summarize the results obtained above and define the rules for transforming a string into a number and, vice versa, a number into a string, with reference to a given dictionary.

Rule #1: Transform a string into a number.

1. Define the dictionary and the elements of the dictionary. The elements in the string must only be composed of any combination of elements of the selected dictionary.
2. Define the NEDE (numerical equivalents to dictionary elements) matrix
3. Apply the following formula:

$$N_0 = a_1 B^{L-1} + a_2 B^{L-2} + ... + a_L$$

Where B is the dictionary's base and α_j is the numerical equivalent for the generic elements of the dictionary, where j=(1,2,...,L) and L is the length of the string.
N0 is the number corresponding to string "$a_1a_2...a_L$", that was searched.

Rule #2: Transform a number into a string.

1. Define the dictionary and the numerical equivalents.
2. Divide number N_0 for the base B of the dictionary, considering both the result of the division and the remainder of the division; in each subsequent division replace the dividend with the result of the previous division, and continue in this way until the result of the division is zero. This happens after L steps, where L is the length of the string.
3. If during execution of the steps described in point 2, you find a remainder equal to zero, replace the result with (result -1) and the remainder 0 with the base B of the dictionary.
4. Starting from the last remainder, create a string composed by the remainders of the divisions.
5. Consider the remainders of the divisions as the numerical equivalents to the dictionary's elements and replace these numbers with the equivalent elements of the dictionary, using the "NEDE" table.


We train in Latin America -
ISTQB Certified Tester Training
contact@bicorp.biz
© iStockphoto

The resulting string is the one that was searched and corresponds to number N_0.

5. Software implementation of the rules

I have prepared two Perl scripts that implement both processes described. Rule #2 described above is suitable to be implemented in a recurring way using Perl (or your favorite programming language), but I prefer to strictly follow the process described to be sure that it will work correctly.
All the scripts can be downloaded from link: http://www.bertadanilo.netsons.org/, or directly from: http://www.bertadanilo.netsons.org/Upload/perl_TestingExperience.zip.

Another Perl script was built with the purpose of generating the mathematical-equivalent function of some software routine. I took as an example the (very simple) routine that produces as output the reverse of the string it has taken as input.

All these scripts were prepared with the sole purpose of providing a "proof of concept". In the following, there is a brief explanation of how they work. All the scripts are released as is, under GPL License.

Script #1. This script, called `StringToNumber.pl` implements rule #1. It takes as input a string and a dictionary in the form of an array of chars separated by spaces, and returns the numerical equivalent to the input string, as per rule #1 explained above. To run the script, from DOS or *IX system, you need to run the following command:

```
perl StringToNumber.pl <input
string> <dictionary>
```

For example:

```
perl StringToNumber.pl ABECDEADD
"A B C D E"
```

which returns the number `637549`

You can avoid having to explicitly call the "`perl`" interpreter before the script name if you give the execution permission (`chmod +x StringToNumber.pl`) on *IX system to the script. In Windows, it depends on how your system is configured. I will not bother you any further with these details. A DOS script called `runStringToNumber.bat` could also be used to run the script (avoiding having to bore yourself with writing the dictionary, which is usually fixed), passing only the string input parameters, as follows:

```
runStringToNumber.bat  ABECDEADD
```

Script #2. This script, called `NumberToString.pl` implements rule #2. It takes as input a number and a dictionary in the form of an array of chars separated by spaces, and returns the numerical equivalent to the input string, as per rule #2 explained above. To run the script, from DOS or *IX system, you need to run the following command:

```
perl NumberToString.pl <input
number> <dictionary>
```

For example:

```
perl NumberToString.pl 637549 "A
B C D E"
```

which returns the string: `ABECDEADD`. The same considerations as for script #1 are valid for this one. A DOS script called `runNumberToString.bat` could also be used to run the script (avoiding having to bore yourself with writing the dictionary, which is usually fixed), passing only the string input parameters, as follows:

```
runNumberToString.bat  637549
```

Script #3. This script, called `EvaluateFunction.pl` implements both of the rules. It takes as input a dictionary in the form of an array of chars separated by spaces, and returns the table of the results of the function that should be used to plot the Cartesian diagram of that same function. The function is written in the code.
In detail, to use the script you will need to:

- Personalize the function to generate the data in order to plot in the main routine of the program. As a default, the function is the "string reverse" that produces as output the reversed input string. For example: ABC becomes CBA.
- Call the program as follows:

```
perl EvaluateFunction.pl <dic-
tionary> <number of point>
```

The output will be a table of data (separated by tabs) with <number of point> points with the x input string and y output string and equivalent numerical value of the software function.

For example, the output of command

```
perl EvaluateFunction.pl "A B C
D E" 10
```

will be a table of 10 values as follows:

X num	Y num	X num	Y num
1	1	A	A
2	2	B	B
3	3	C	C
4	4	D	D
5	5	E	E
6	6	AA	AA
7	11	AB	BA
8	16	AC	CA
9	21	AD	DA
10	26	AE	EA

Table 4: Example

6. The plot of the "string reverse" function

The first column is the X input numerical equivalent of the input strings in the third column, while the second column is the Y output numerical equivalent of the output strings in the fourth column (that is the real output of the reverse string function, or whatever function you decide to call).

These scripts were tested on Perl version v5.10.0 built for MSWin32-x86-multi-thread for the Windows Vista operating system, and Perl version v5.8.8 built for i486-linux-thread-multi on debian version (Kurumin). These are not really intended to control the formal validity of the inputs used (wrong numerical format, wrong dictionary format, and so on); therefore please take care to use the right format, otherwise the result cannot be guaranteed.

Now, at last, we will attempt to plot a Cartesian diagram of a software function that works with a list of symbols (strings). To keep the task simple, we have decided to consider the reverse string function. Using the command

```
perl EvaluateFunction.pl "A B C
D E" 100 > out.txt
```

we have in file `out.txt` a table of 100 points that can be used to plot the "reverse string" function. For this purpose, you may use your favorite spread sheet or any other software you usually use to draw graphs.

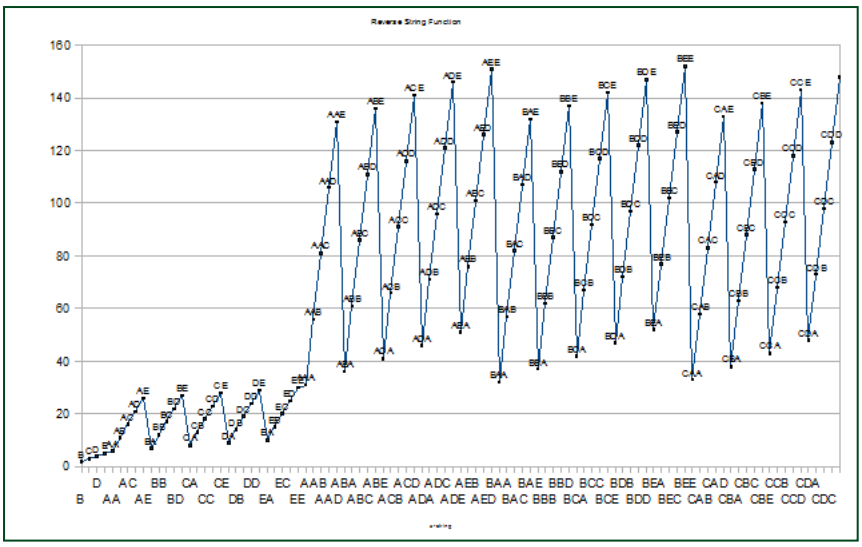
The plot looks like Graph 1.

The Y axis shows the numbers and not the Y-string value, to make the graph easier to manage. Considering that the function is simply a "reverse string" it is easy to imagine the corresponding string value of the Y interval number.
A graph with more points (in this case 500 points), but with fewer details, is shown below.

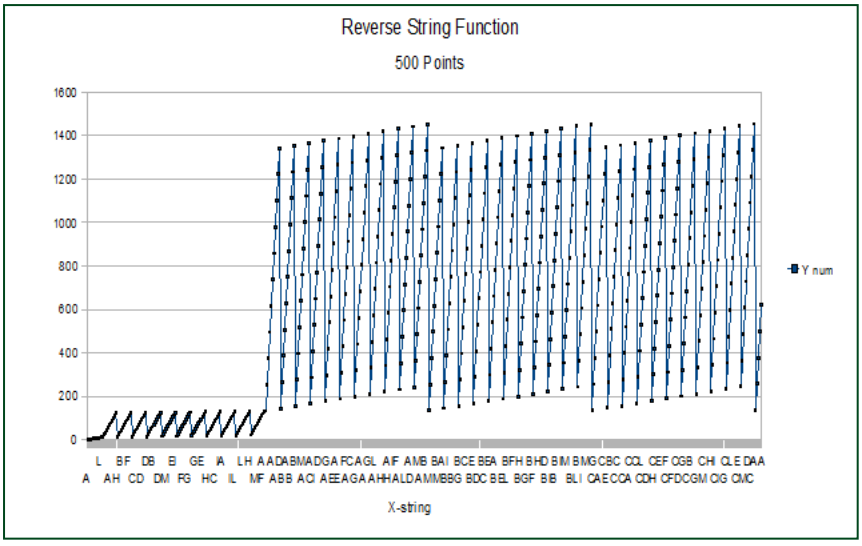
The graph with 10000 points looks like Graph 3.

7. Conclusions: What does all this have to do with software testing?

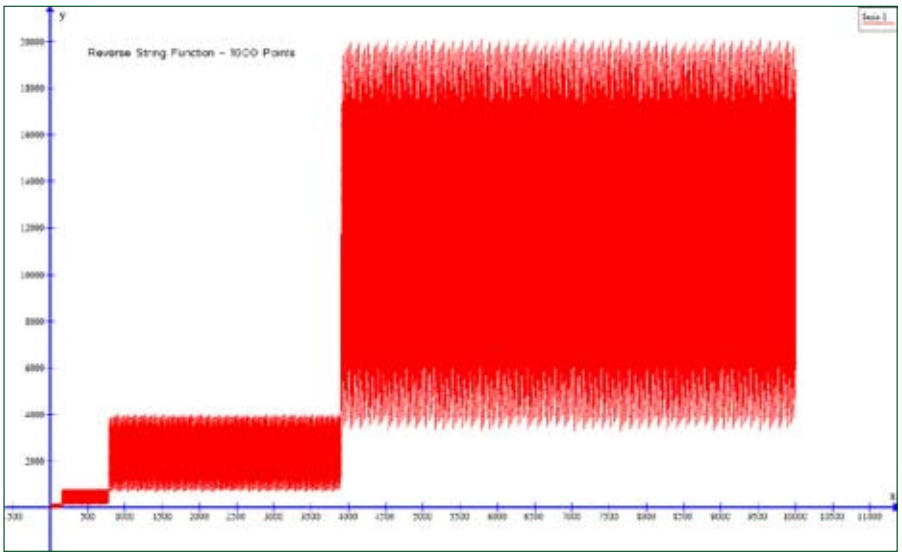
So, we have probably found a method to "reduce" a software routine into a mathematical function. Clearly, what we have described can "easily" be further generalized.
What does all this have to do with testing? At the current moment absolutely nothing; the main idea, as you can guess, is the following: if we know what the mathematical equivalent function behind the software routine under test is, we can easily "test it" under a (small or large) sample of input testing points (the "X" axis).
Now, the problem is that nobody (also, and in particular, no one who wrote the software) knows of the function. I think that nobody knows the profile of the "reverse string" function graphics presented above, or that the "reverse string function" is a well-known and



Graph 1: Shape



Graph 2: Shape



Graph 3: Shape

widely used feature in a lot of software programs.

So, one might ask what this method could be useful for? It could be useful in regression testing, in particular for the testing of single software components and for the test of (probably huge) amounts of database data, especially for systems that cannot be easily accessed, for example, systems who are located in different, non-integrated environments and for which access to the data for the software under test can only be produced only by extraction (e.g. in CSV format). In regression testing you can use the information collected during the "first test run" to clearly define the expected behavior of the function under test. You can therefore transform it into a numeric function (one or more functions) and keep it as reference for the other entire regression test or retest. The advantages are:

a. You collect input and expected output data in numerical format. Knowing the numerical input and output data, you can at any time recalculate the inputs and outputs needed for the software routine (see NumberToString.pl proof of the concept script). Clearly, you must also save the dictionary used in the transformation.

b. After you have run the software on the input data, you can transform the real output data into a table of numbers and draw the "real function". This function must then be compared with the "expected function" (i.e. performing a point-by-point comparison between the two functions). If any differences between the two are found, these can be analyzed. The operation can be easily done using any mathematical software that can draw graphics.

For example, to keep things simple, if you enter users onto a system and would like to verify that all the users' fields are correctly loaded onto the database, you can proceed as follows:

1. Prepare the file with the data to be loaded.
2. Transform all the columns for which you want to verify correct entry into the expected functions.
3. Load the data using the software under test
4. Extract the loaded data from the database and transform it into a function (i.e. a real function, which expresses the real software behavior); then compare the two functions.

This has been done for the first time. For the following test, you can automatize the process. Starting from the input and output numbers (which you saved in the first run), you proceed as follows:

a. Extract the input strings from the input numeric data and from the dictionary.
b. Run the software on the inputs and get

the outputs; transform the outputs into numbers and draw the function(s).

c. Compare the expected output (numbers) with the real output (numbers) and find the differences (if any).

Clearly, there will be no major differences if we compare this method with a more traditional "string comparator". With this method, you will probably not have to develop new "comparator software" to perform a particular comparison, because all you need is software that can draw graphics starting from a table of (x;y) values (or also for more variables).

Acknowledgements

I'm grateful to my friend and colleague Gianmaria Gai for the help he has given me when I wrote this article: he also suggested that I include some references. The problem is that there aren't any. The subject occurred to me one day when I was reading a (very boring) functional analysis for some software. I thought along the lines: "Wouldn't it be nice if we could reduce a big analysis into some mathematical functions! How we can do this? "...
If any reader knows about some other articles and/or books on similar methods, please let me know.

Biography

Berta Danilo graduated in Physics at Turin University (Italy) in 1993 and first started to work as a Laboratory Technician, but soon he switched to the software field as an employee of a consultancy company working for a large number of customers. Throughout his career, he worked in banks as Cobol developer and analyst, for the Italian treasury, the Italian railways (structural and functional software test), as well as for an Italian automotive company, the European Space Agency (creation and test of scientific data files for the SMART mission), telecommunication companies, and the national Italian television company. This involved work on different kinds of automation test projects, software analysis and development projects.
With his passion for software development and analysis – which is not just limited to testing – he has written some articles and a software course for Italian specialist magazines.
His work gives him the possibility to deal mainly with software testing; he holds both the ISEB/ISTQB Foundation Certificate in Software Testing and the ISEB Practitioner Certificate; he is a regular member of the British Computer Society.
Currently, he works as Test Manager for an Italian Editorial Company.

Database Testing - Testing The Backend

by Gulshan Bhatia

Database systems play an important role in nearly every modern organization. The ultimate objective of database analysis, design, and implementation is to establish an electronic data store corresponding to a user's conceptual world.

Functionality of the database is a very critical aspect of application's quality; problems with the database could lead to data loss or security violation, and may put a company at legal risk depending on the type of data being stored. Applications consist of a database and improving quality of data in an organization is often a daunting task.

A database should be evaluated throughout the database development lifecycle to prepare a quality database application. Data in a database may be input from and displayed on a number of different types of Systems. Each of these types of systems has unique system limitations, which may dictate how data should be formatted in your database. A database should be evaluated based on the factors such as data integrity, consistency, normalization, performance, security and very important - the expectations of its end users.

We can attempt to maintain the quality standards in a proactive way and it involves surveying the customer to determine their level of satisfaction with the product so that potential problems can be detected early in the development process.

Once the development is over we can ensure that the database conforms to predefined standards and guidelines. It compares the required features or attributes (problem domain) with the results of development (solution domain). When deviations from the problem domain are found, they are resolved and the process is modified as needed.

The database design process is decided by finding the requirements and needs of the end user. Uncertainty about understanding the requirements can be reduced only after significant analysis and discussions with users. Once the user requirements are clear, the process of behavior implementation consists of the design and construction of a solution domain following the problem domain. Because of the difficulties associated with the changing requirements, the database developer must attempt to develop a database model which closely matches the perception of the user, and deliver a design that can be implemented, maintained and modified in a cost-effective way.

Diagrammatic representation using entity-relationship diagrams, object models, data flow diagrams, allows the information described in a visual format in a meaningful way.

Database testing is one of the most challenging tasks to be done by software testing team. **A Database Tester**, by understanding the referential integrity and database security, and by having a good grasp on the various technologies and data formats used to transfer and retrieving the data from the database, can test database to avoid problems. Testing should be included in various phases of database development lifecycle. The cycle typically consists of several stages from planning to designing, testing and deployment.

In the first phase of database development process, requirements are gathered; checklists can be used as part of the evaluation process for the database specification.

After gathering requirement and understanding the need for the database, a preliminary list of the fields of data to be included in the database should be prepared. We should have complete information about what information is required by the client and what type of fields are required to produce that information. Next determine if the logical data model is complete and correct. Confirm the design with the business community in a design review process. Create a logical Entity Relationship Diagram (ERD) to graphically representthe data store. Determine if the data model is fully documented (entities, attributes, relationships) Attributes have correct datatype, length, NULL status, default values. General discussion of business rules to be enforced by database level constraints should be carried out e.g.

- Not null constraints
- Check constraints
- Unique constraints
- Primary key constraints
- Foreign key constraints

Business rules to be enforced by triggers and procedures should be discussed along with the Business rules to be enforced by application code. After this the normal forms should be tested with the help of test data. Testing physical database design includes testing table definitions, constraints, triggers and procedures. Black Box testing techniques like Boundary value analysis can be used. We can test the table definition by testing the column definitions and constraints that have been imposed:

Database constraints can be checked as follows:

1. Primary Key- Write Test Case to insert duplicate value in Primary Key column.
2. Insert record to violate Referential Integrity Constraint.
3. Delete record to violate Referential Integrity Constraint.
4. Insert NULL value in NOT NULL column.
5. Insert values to violate check constraints by inserting values outside input domains.

SOFTWARE & SYSTEMS
QUALITY CONFERENCES
United Kingdom | 2008

Tutorials

On Wednesday 1st October, we are holding a series of tutorials which complement the presentations from the conference. These are being held at The Institute of Mechanical Engineers.

SQC conference 2008 | Monday 29th September and Tuesday 30th September | QEII Conference Centre London

Never too **busy?**

The role of testing in improving productivity.

SQC Conference 2008

We're looking to create a real buzz in 2008 with a fresh, new venue at the QEII Conference Centre. And SQC UK will be a hive of activity this year with some fascinating speakers for what promises to be a challenging, and potentially controversial, conference theme: never too busy – the role of testing in improving productivity.

Everyone needs to pull their weight in an organisation. If a worker bee doesn't contribute to a hive, the entire colony is at risk (and no-one gets any honey). The same can be said of organisations whose departments aren't contributing effectively to the business (although honey levels aren't affected).

With insight from many industries, including financial services and enterprise management, SQC UK will tackle this sticky subject head on and discuss exactly what the testing community can do to boost productivity in the business space.

Be the bee's knees in your company...

Venue

Mountbatten Suite and
Elizabeth Windsor Room,
5th floor, QEII Conference Centre, London

Dates

Monday 29th September and
Tuesday 30th September

Delegates

Practitioners and business delegates:

1 day: £400 + VAT per person

2 days: £750 + VAT per person

Tutorial Day - £350 + VAT per person

Exhibitors

There are a number of exhibitor packages available, including sponsorship deals.

Organised by

SQS
SOFTWARE QUALITY SYSTEMS

For more delegate or exhibitor information, please contact the conference team.

Phone: +44 (0)20 7448 4647

Email: uk@sqs-conferences.com

Website: www.sqs-conferences.com/uk

For relational databases queries are written using SQL. We can test database queries by identifying different situations and data values. SQL conditions can be tested using Decision Condition Coverage:

a. Select statements use conditions in Where Clause and Having Clause for Group By columns.

   i. Conditions written using AND logical operator requires (T,T), (T,F), (F,T) outcomes for two operands to be tested.
   ii. Conditions written using OR logical operator requires (F,F), (T,F), (F,T) outcomes to be tested.

Testing requires that every condition affecting the result takes all possibleoutcomes at least once.

b. Testing SQL statements involving NULL values.

   i. Requires Testing conditions with each operand in the condition taking NULLvalue.
   ii. For a Group By Clause, NULL values have to be considered.

c. For subqueries, include test cases to return zero and more rows.
d. SQL statements like Update, Insert, Delete also need to be tested for conditions.

Apart from testing the table definitions and SQL statements a database tester should test the triggers, procedures and functions. These objects can be unit tested by finding various paths of execution in the code and functionality can be tested by executing the code - providing required inputs and checking the output generated.
Lets see how can we design test cases to test table definition i.e. column definitions and constraints. Refer to 'Item' table storing details of items in stock. Details of sales orders placed for various items are stored in another table 'Sales'. The table definitions are as follows:

ITEM Table

Col Name	Datatype	Size	Constraints
Itemcode	Char	4	1001 to 1555 Primary Key
Description	Varchar	20	NOT NULL
Price	int		>0
QOH	int		
ROL	int		QOH>ROL>0

SALES Table

Col Name	Datatype	Size	Constraints
Order_ID	Char	4	Should begin with O
Item_code	char	4	References Items Table
QTY	int		>0
Order_date	datetime		
ROL	int		QOH>ROL>0

To test the constraints, we can design test cases as follows:

ITEM Table
Test Case ID: TCcheck _itemcodePK
Objective: To evaluate Primary key constraint on Item code in item table.
Description : Insert two records with I001 as Item code.
Expected Result : second record should not be saved in database.

Test Case ID: TCcheck _itemcode1
Objective: To evaluate check constraint on Item code in item table.
Description : Insert a record with I001 as Item code.
Expected Result : record should be saved in database.

Test Case ID: TCcheck _itemcode2
Objective: To evaluate check constraint on Item code in item table.
Description : Insert a record with I555 as Item code.
Expected Result : record should be saved in database.

Test Case ID: TCcheck _itemcode3
Objective: To evaluate check constraint on Item code in item table.
Description : Insert a record with invalid Item code as I000.
Expected Result : error message should be displayed.

Test Case ID: TCcheck _itemcode4
Objective: To evaluate check constraint on Item code in item table.
Description : Insert a record with I556 as Item code.
Expected Result : error message should be displayed.

Test Case ID: TCcheck _Description1
Objective: To evaluate NOT NULL constraint on description column in item table.
Description : Insert a record with no value for description column.
Expected Result : error message should be displayed.

Test Case ID: TCcheck _price
Objective: To evaluate check constraint on price column in item table.
Description : Insert a record with price=-10.
Expected Result : error message should be displayed.

Test Case ID: TCcheck _delete
Objective: To evaluate referential integrity constraint on item code column in item table.
Description : insert a record in sales table with item code I001. Delete record from item table where item code=I001.
Expected Result : error message should be displayed.

SALES Table
Test Case ID: TCcheck _itemcode
Objective: To evaluate references constraint on itemcode column in sales table.
Description : Insert a record with item code not existing in item table.
Expected Result : error message should be displayed.

Test Case ID: TCcheck _orderID
Objective: To evaluate check constraint on Order_ID column in sales table.
Description : Insert a record with Order_ID as 1001.
Expected Result : error message should be displayed.

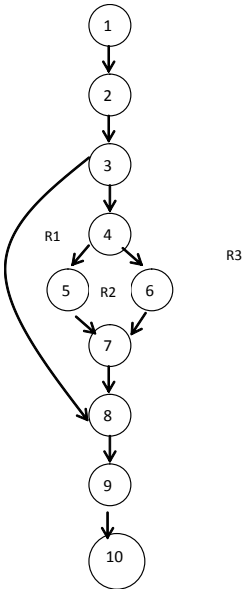
Now consider a requirement for displaying details of all the items for which qty ordered is>10. We can find the required details by writing SQL query:

```
Select item_code
from sales
where qty>10
```

To test this query we can prepare a test sales table having records with qty>10. When executed, this query should return all the matching records. Similarly we can check the query by preparing a table not having any matching records.

We can unit test the triggers, procedures and functions by finding various paths of execution in the code. Consider the following PL/SQL block:

```
1:  Declare
    Bonus number:=0; --required values assumed
    Sales number:= 6000;
    Target number:= 5000;
2:  Begin
    differ:=sales-target;
3:  if differ>0 then
4:      if differ>100 then
5:          bonus:=1000;
6:      else
          bonus:=500;
7:      end if;
8:  end if;
9:  update salesman
    set incentive=bonus
    where empcode=1;
10: End;
```



Control Flow graph

No. of Edges: 11
No. of Nodes: 10
No. of Regions: 3
No. of Predicate Nodes: 2
Cyclomatic Complexity : 3

There are 3 independent paths. We can design the test cases to test each path using statement coverage and decision-condition coverage.

Apart from testing constraints, triggers and procedures; database security also needs to be checked. We need to see how database can be made secure. Test data plays very important role in database testing process. Database developers and administrators spend a lot of time in creating test data sets to examine database performance. Data generation tool can be used to avoid this and automatically create test data. Tool can also be used to stress test database server by creating a continuous set of requests to the server.
Thus all the factors must be considered to improve quality of database and in turn improving the entire business process.

Biography

Miss Gulshan Bhatia currently works at WebTek Labs Pvt. Ltd. - a leading Software Testing company in India. She has a broad background in the field of Information Technology and Software Testing. Gulshan is involved in conducting Software Testing Trainings at corporate and public level; designing and updating the courseware; providing consultancy for Software Testing projects using Manual Testing Techniques and Automation Tools. Her interests include Database Testing, Web Application Testing and Telecom Testing.

iSQI®
International Software Quality Institute

QAMP®

Quality Assurance Management Professional

Best prepared.

Tailor-made Modular System

Three certificates:

1) Requirements Engineering (IREB® CPRE, FL)
2) Software Test (ISTQB® CT, FL)
3) Work environment specific module of choice:
   - Test Management (ISTQB® CT AL-TM)
   - Configuration Management (iNTCCM®)
   - Project Management (iSQI® CPPM)
   - Software Architecture (iSAQB® CPSA)
   - Secure Software Engineering (ISSECO®)

   (Other certificates need to be recognized by iSQI.)

Experience evidence:

At least two years of practical experience in software quality assurance management required. Annual re-certification.

www.qamp.org

Strategies for Testing Telecommunication Services

by Bárbara Martínez , Jesús del Peso and Lourdes Calvo

For a telecommunications operator, the final outcome of its activity is a series of services to be offered to the public. The quality of these services and their perception by end users is essential for the operator's business model.
This article covers the issue of the design and execution of large test campaigns for telecommunication services, and does so from a pragmatic point of view. In addition, some basic design principles for the design and construction of general testing tools for telecommunication services are discussed.

Introduction

First of all, it is necessary to define precisely what is meant by telecommunication service. One possible definition of this term is found in the Telecommunications Act of 1996 [1]:
"The offering of telecommunications for a fee directly to the public, or to such classes of users as to be effectively available directly to the public, regardless of the facilities used."; where "Telecommunications", in turn, is defined as "the transmission, between or among points specified by the user, of information of the user's choosing, without change in the form or content of the information as sent and received."

Telecommunication services are very complex types of service. Their complexity is due to both structural and functional reasons. For example, from the structural point of view, telecommunication services are composed of a high number of elements. These elements can be of very different types:

- Hardware elements which include: user equipment and terminals (such as mobile phones, fixed phones, personal computers, PDAs or TV set-top boxes) and the hardware elements that compose the core network, (such as routers, switches, gateways, radio stations, etc)..
- Software elements which include: the programs and software systems that control hardware elements, which can be of very different types (such as clients, servers, databases, directories, codecs, etc)..

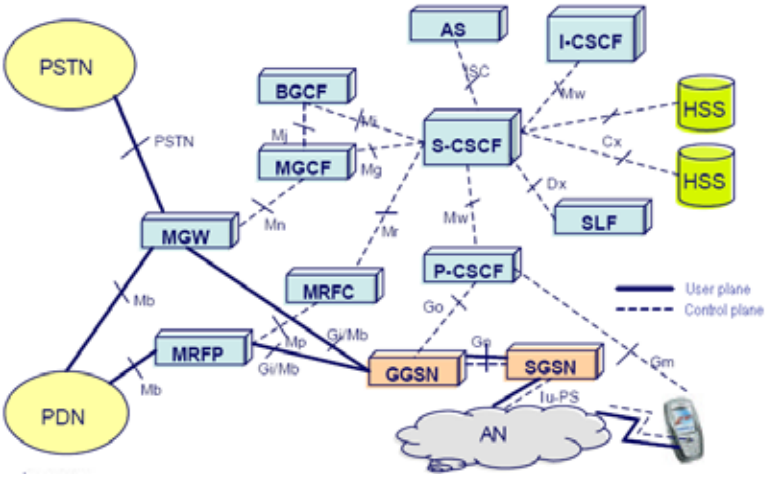


Figure 1: Example of architecture of a telecommunication service (IMS).

- Several sets of logical elements, like the communication protocols through which these elements communicate, or the interfaces and reference points through which this communication takes place.

Therefore, the complexity of telecommunication services comes mainly from the multiple interactions established between the different elements that compose the whole system. In addition, the single elements usually have behaviours complex enough individually.
However, this complexity is not only due to these internal elements. It is also due to the fact that they are public services, and therefore they must be able to interoperate with other similar services of different operators in order to provide a whole service.

Testing Telecommunication Services

Due to the high complexity of telecommunication services, the phase of certification and testing becomes crucial. However, testing of telecommunication services is also a very complex task for several reasons:

- It is necessary to model the behaviour of some of the elements of the service and frequently one or more final user also have to be simulated in order to be able to perform complete tests. In other words, it is necessary to build behavioural models for the whole service and for some of its components.
- It is also necessary to verify multiple conditions on some of the elements and equipment composing the service.
- The number of different situations and scenarios that have to be tested is also very large.
- It is necessary to test the new releases of the service efficiently. Therefore it is very important to be able to ensure reproducibility of the testing scenarios and to guarantee fast execution and analysis of the test campaigns.

Therefore, test campaigns are usually very large, and are composed of a large number of individual tests. Test parameterisation is usually very complex and great flexibility is required for test definition, configuration and execution.

One important problem to be solved is connected with the reutilisation of code: since a telecommunication service is composed of many different elements, it is necessary to combine various independent test libraries easily to build specific test tools for new services.

Another problem that must be tackled is the provision of tests: the execution of telecommunication service tests requires provisioning complex configuration information to achieve a successful service simulation. Before executing any test, it is necessary to provision it, which means to configure each one of the elements that compose the telecommunication service. This implies, for example, the modification of databases or directories and even the manual configuration of some equipment. After execution of the tests it is also necessary to undo the previous provision in order to leave

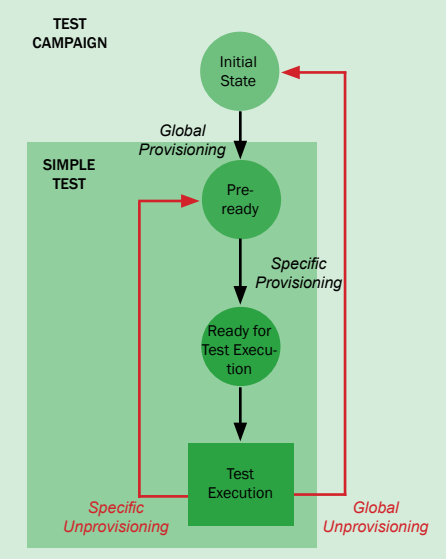


Figure 2: Execution of tests including automated service provision phases.

the system ready for the next test execution, as described in Figure 2.

To be able to efficiently carry out tests for telecommunication services, we have proposed —and are currently using at Telefónica I+D— a solution based on two fundamental principles:

- Using standard languages and tools for the definition and implementation of testing systems.
- Modelling complex behaviours of service through high level models and formalisms, which allows us also to highly automate both the definition and the execution of tests.
- Using data-driven models of testing to increase code reusability and to simplify test development and management.

Using Testing Standards: TTCN-3

TTCN-3 (*Testing and Test Control Notation version 3*) [2] is a standard test definition language, created by ETSI (*European Telecommunication Standards Institute*) [3]. TTCN was initially designed for the specification and execution of telecommunications protocol tests. However, since *version 3* it supports virtually any kind of tests, including software testing. Therefore, TTCN-3 is beginning to be popular not only in telecommunication testing environments, but also in other sectors such as consumer electronics, automotive and software industries.

One of the main reasons for adopting the TTCN-3 standard is the reutilisation of test code and libraries. As telecommunication services are composed of multiple types of elements and equipment, it is fundamental to be able to combine several test components to quickly build new testing systems for new telecommunication services.

TTCN-3 defines a test language which can be compiled for multiple software and hardware platforms; it has been defined with code reutilisation as one of its main design goals. TTCN-3 allows this reutilisation through a set of standard interfaces — which make it possible to build test adaptation libraries and reuse them in different systems—, and the concept of ATS (*Abstract Test Suite*), which contains the logic defining the tests, expressed in a system-independent way, and is composed of several reusable *modules*.

In addition, some standardisation organisations, such as ETSI or 3GPP (*3rd Generation Partnership Project*) [4] are adopting TTCN-3 as a way for defining conformance test suites for their specifications. This is the case, for example, for the conformance ATSs of SIP (*Session Initiation Protocol*) [5], IPv6 [6] or WiMAX [7] by ETSI.

Modelling Complex Behaviours for Testing

Despite the advantages of using TTCN-3 as testing language, it is not enough to define and execute complete telecommunication services tests. The TTCN-3 language —as many other testing and programming languages— is a reasonably low-level language. This means that it allows to define tests to a very high level of detail, but it is difficult to define complex behaviours: using TTCN-3 every single step of the execution of tests must be coded, but the costs of doing this are unaffordable in terms of time and effort in our testing scenarios.

To achieve our goal, a more friendly and scalable formalism is needed to model telecommunication services behaviours.

In addition, TTCN-3 test campaigns involve the execution of several test cases. These test cases may be parameterised to allow the execution of different types of tests to simulate a set of testing scenarios of interest. As mentioned in the previous sections, one of the main characteristics of testing telecommunication services consists of the large-size test campaigns: they usually have several hundred, even thousands, of different tests which have to be implemented and executed.

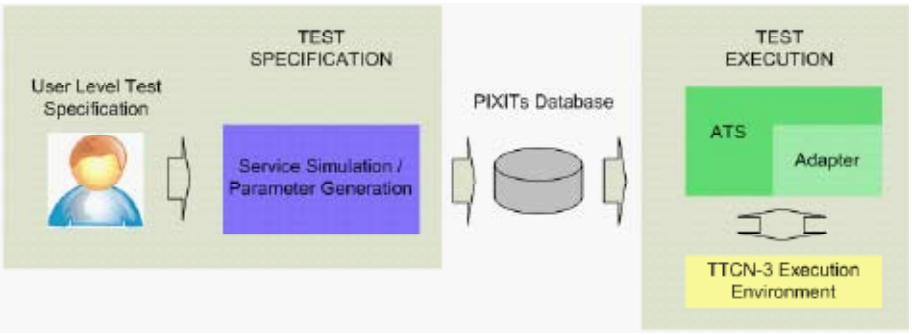


Figure 3: Phases of the definition and execution of complex telecommunication services tests.

To deal with the execution of large test campaigns and with the high complexity of service behaviours to be tested, we have proposed to separate the execution of tests from their definition, as shown in Figure 3.

We propose to make the test cases as general as possible. Test cases must also be highly parameterised. Our main goal is to control test behaviours through parameterisation as much as possible. Therefore, we distinguish two types of parameters:

- *Module level parameters*, which are common to the whole test campaign and contain general configuration information – these parameters are defined as *modulepars* in TTCN-3 terminology–.
- *Test case level parameters*, which are specific to each of the individual tests of the campaign – defined as the formal parameters of test cases –.

Therefore, test campaigns are defined by a set of general configuration parameters and one or more databases, which contain the specific parameters that define each of its individual tests.

We propose to extract the main part of the complexity of behaviours from TTCN-3 code to external tools, which help us to specify tests and perform complex simulations.

Therefore, we have decided to make test cases as simple and general as possible and to parameterise them completely. This means that TTCN-3 test cases contain the code which is necessary for the execution of quite generic tests, but the main part of the behaviour of the test is driven by test parameters.

This implies that the number of parameters

needed for each test case is now very high – there could even be hundreds of them –. However, as the number of parameters of the test cases increases, it becomes more difficult to assign them their corresponding values. Sometimes these parameters are even too complex to be specified manually.

For these reasons we have decided to use a set of external tools that allow us to:

- Use a high level model for the specification of tests, which makes it possible to easily define complex behaviours and service simulations.
- Automate the codification of parameters.

These external tools separate the specification of tests from their execution and contain the main part of the logic that models the behaviour of services under test, using, for example, rule-based systems – in which test behaviours can be specified declaratively as sets of rules –. The external tools also offer a simplified model and specific user interfaces for the specification of large test campaigns, integrating the use of databases of test cases. They are also responsible for the generation of the provision information and parameters and for the automation of provision execution.

In Figure 3 we can see how the specification and execution of tests is separated: there exists a first phase in which the specification of tests takes place using a set of external tools. This phase produces as its result one or more databases which contain the parameters that define each one of the individual tests of the test campaigns. The second phase consists of the execution of the previously simplified test cases using the parameters contained in test databases, which usually involve several provision steps.

TTCN-3 data-driven testing

TTCN-3 development is mainly related to the testing implementation process.
As a result of the TTCN-3 specification activity, an ATS (Abstract Test Suite) is obtained. This ATS is transformed into an ETS (Executable Test Suite) after compiling, and is used in the testing execution process.

If the traditional TTCN-3 development methodology is used, a testing design modification implies repeating the TTCN-3 specification and compilation activities. In this situation, the following disadvantages are found:

- Development costs: every time a design modification is done, a person with TTCN-3 skills is needed to update testing implementation.
- ATS version control: the number of ATS versions increases significantly; debugging tasks become more complex, and the probability of error increases.

With a data-driven approach, the ATS is designed to read all values used during the execution from a database.
As shown in Figure 4, the ATS fetches a complete message from each field in order to reduce the number of database fields; normally this message is a complex PIXIT (Protocol Implementation eXtra Information for Testing), which means a PIXIT defined by a structured type.

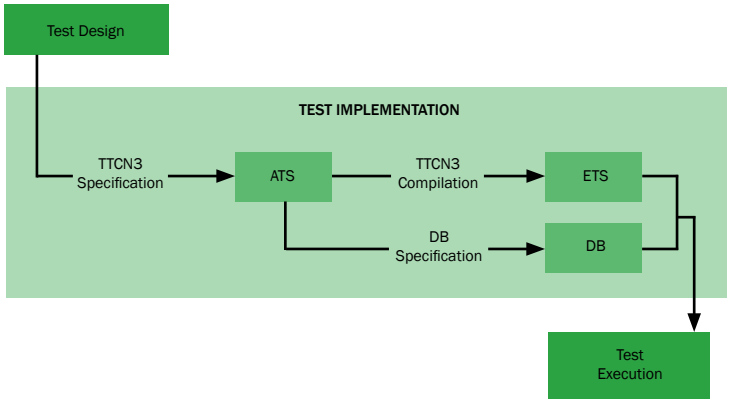


Figure 4 : Data driven approach and TTCN-3.

Applying this philosophy, a testing design modification has a much lower impact in testing implementation. It is enough to modify a database, which could be an Excel file, for example.
The data-driven concept applied to TTCN-3 provides the following benefits:

- Reduction of development costs: it is not necessary to undertake a new TTCN-3 development each time the test design is modified.
- Version control and defect probability minimisation: once the initially developed ATS is debugged, it is not necessary to make any further ATS versions with each test design modification and its debugging.
- A person without TTCN-3 skills can specify the tests.
- Testing specification can be automated, for example by visual basic macros, set of rules, etc. This makes the specification task easier, and reduces the amount of repetitive work and the probability of error.
- Lower impact of SUT (System Under Test) updates in test implementation: when there is a design modification in the SUT, it is enough to modify the testing database rather than adapting the ATS.
- Reusability of test cases: it is possible to validate different services with the same test case by applying small configuration changes.

Using the traditional methodology, as shown in Figure 5, a test case is defined as a pre-determined sequence of sent/received messages, with some condition statements. This kind of test case implies that each time the sent/received messages sequence is changed, a new TTCN-3 development is needed.

In a test case developed with the data-driven approach, the messages sent and received, the number of iterations and the timers between iterations are read from a database, as described in Figure 6. Messages are usually represented by complex PIXITs, in order to minimise the number of PIXITs handled. With this methodology all protocol messages are defined in the ATS, and subsequently it is possible to configure any send/reception sequence defined by the test design using a database. Therefore, only one test case is required, which consists of a loop with n-iterations 'send/reception', configurable through a database.

It is possible that there are test scenarios where data are unknown at specification time, and therefore they must be obtained during execution time. For these scenarios, an import/export mechanism can be implemented.

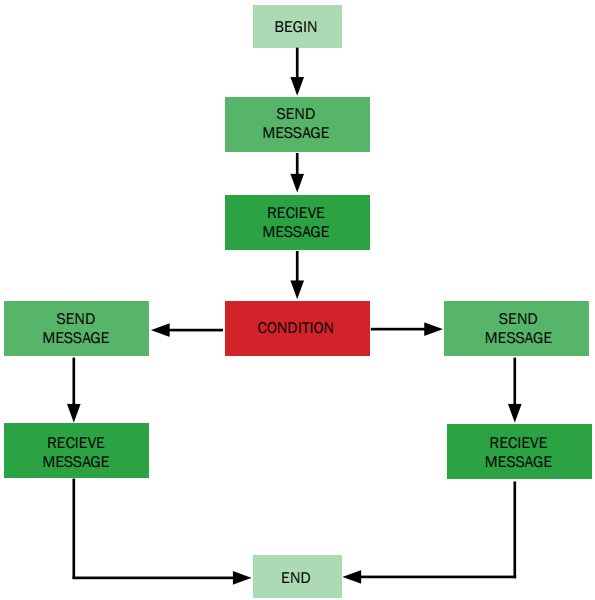


Figure 5 : Traditional TTCN-3 test cases.

Previously received data is exported to a data buffer, and is imported to the data buffer to be sent. This can be configured in the database by means of import/export flags.
If any data calculation is needed during execution, then the appropriate keyword is defined to be used in the scenario's definition in the database.

Conclusion

The complexity and large dimension inherently associated to telecommunication services testing imply special requirements to the test-

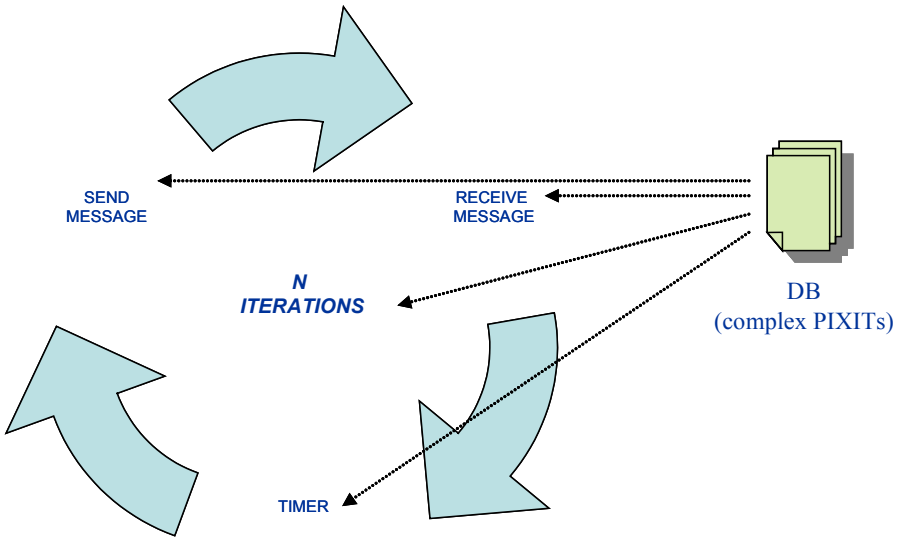


Figure 6: Data driven TTCN-3 test cases.

ing process of a telecommunication operator.

Within this context, the main prerequisites in the testing process are: necessity of simulating the service performance, flexibility and repeatability of the scenarios' configuration, and efficiency and reproducibility of test campaigns for new releases of the service.

To deal with these requirements, Telefónica I+D has developed a testing methodology based on two basic principles: use of standard languages and tools for the definition and implementation of testing systems, and replicating complex service behaviours through high-level models and formalisms.

Telefónica I+D has chosen TTCN-3 to implement the testing systems because it has been specifically designed for testing and certification, and it allows the reutilisation of test code and libraries. In order to make the testing process based on TTCN-3 more flexible, external tools (rule-based tools) have been introduced to separate the specification of tests from the execution. These tools reproduce the service by using behavioural models.
The combination of TTCN-3 testing environment with the service's behaviour modelling has been successfully achieved by introducing a new approach: TTCN-3 data driven testing.
This approach has several advantages in terms of minimising resources and costs, compared to a traditional solution.
Telefónica I+D is currently using this methodology to certify telecommunication services, and it has obtained positive results by improving considerably the efficiency and quality of the testing process.

References

[1] "Telecommunications Act of 1996," Pub. LA. No. 104-104, 110 Stat. 56, 1996.

[2] "TTCN-3 Home page"; http://www.ttcn3.org/.

[3] "ETSI"; http://etsi.org.

[4] "3GPP"; http://www.3gpp.org/.
[5] "Conformance Test Specification for SIP (IETF RFC 3261)"; http://webapp.etsi.org/WorkProgram/Report_WorkItem.asp?WKI_ID=19801&curItemNr=8&totalNrItems=12&optDisplay=12&titleType=all&qSORT=HIGHVERSION&qETSI_ALL=&SearchPage=TRUE&qETSI_STANDARD_TYPE='TS'&qINCLUDE_SUB_TB=True&qRAPTR_NAME=Fischer&qRAPTR_ORGANISATI.

[6] "TC MTS-IPT: IPv6 Testing an eEurope Project"; http://www.ipt.etsi.org/deliverable.htm.

[7] "Open IP Testing Library"; http://www.ipt.etsi.org/STF295-ph1/home.asp.

Biography

Jesús del Peso graduated in Telecommunication Engineering at the Technical University of Madrid where he is currently PhD candidate in Computational Mathematics and Artificial Intelligence. He is involved in researching in the field of multiagent systems, semantic technologies and the Semantic Web.
He has applied several IA techniques in several projects during his career as consultant, and for the last four years he has been responsible for the design and development of several simulators and test specification and execution test systems at Telefónica I+D for environments ranging from SS7 (Signalling System number 7) to IMS (Internet Multimedia Subsystem).
He has used TTCN-3 as the basis of his work in testing for the last three years, participating in the last two TTCN-3 User Conferences.

Lourdes Calvo is a project manager at Telefónica I+D in the "Service's Monitoring Platform" Division. She is responsible for projects on testing of mobile services (Telefónica MoviStar). She is currently in charge of managing a team working on the development of testing tools for protocols of mobile communications based on TTCN-3 (testing environment defined by ETSI). She has been also involved in the design, development and application of other testing systems in the area of mobile communication, specially related to Intelligent Network (IN).
She graduated in Telecommunication Engineering at Superior Polytechnic Institute of Zaragoza, (Spain) and made a M. Sc. thesis on optical communication in Denmark Technical University. In 2000 she joined "Testing Environment for Mobile Networks" Division in Telefónica I+D performing tasks related to specification and development of testing to Intelligent Network services.
Currently, Lourdes is also involved in European projects related to testing such as Panlab and PII.

Bárbara Martínez graduated in Technical Telecommunication Engineering at Polytechnic University of Madrid. She has worked as a tester since 1999.
As an ISTQB Foundation Level Certified Tester, she has developed different testing tools using TTCN2 and TTCN3 specification languages.
She has also taken part in the TTCN3 Users Conference 2008, in Madrid.

Control integration and test activities in your project

by Erwin Engelsma

Introduction

I suppose we all know the problems that projects have when they start to integrate products from multiple development teams. Partly finished products that seemed to work correctly in isolation do not work well when integrated. Testers find problems more rapidly than developers can solve them (or even assign them to subprojects), and all of a sudden a project that seemed to run on time gets delayed. Depending on the experience of the project team, testers may get blamed, or may get asked to test less, in order to meet the project deadline. This usually results in bad product quality going to the customer, which will generate a business headache in the future. The problem gets more difficult to handle if the project develops software from several software disciplines and hardware (electronics) or mechanics, or employs several teams. Many different causes may underlie this phenomenon. To mention a few:

- Measurement of project progress in terms of 'phases' completed, like: 'Requirement document completed', 'Design document ready'
- Overly optimistic view of the design complexity
- Designs not clear enough, ambiguous, or not communicated well enough
- Misunderstandings between people from various disciplines
- Overly optimistic planning
- Insufficient or skipped (time pressure!) test 'moments'
- Insufficient synchronization between deliverables from various teams
- Insufficient control over interfaces (lacking formal verification)
- Insufficient design modeling
- Insufficient attention paid to test methodology and environments
- Essential parts of the test environment not developed, or have insufficient quality
- Insufficient information present to act upon mishaps or unexpected events
- High level of cognitive dissonance needed before project management sees problems

So-called 'Agile' or 'Evolutionary' methods help to reduce this problem. Using Evolutionary methods you may find faults quicker (you detect them sooner after the development team introduced them).

Philips Healthcare uses evolutionary methods to develop products and within this context my group applies a method that:

- Controls integration steps based on pressing technical relationships
- Makes visible (graphically) how the project team will integrate partly completed products
- Plans development of test means, methods and environment
- Monitors project progress in terms of what the teams measurably achieved

The graphic overview that displays the needed information we call the 'Master Integration Diagram'.

Master Integration Diagram description

The Master Integration Diagram (MID from now on), makes use of the following elements:

- A **Deliverable** (D) from a development team
- An **Integration Moment** (IM) between deliverables
- A **Test Moment** (TM) for integrated deliverables
- An **Increment Delivery Moment** (IDM) for completed functionality that the project will deliver to a customer

I will show below (see Fig 1) what these elements and the MID look like. I would like to stress that the MID differs from a Gantt chart in that the MID provides necessary technical (or other) relationships, and does NOT plan activities.

In order to make the MID easily creatable (no special tools needed), we use only a few symbols that you can find in Word, and connectors, so we can easily shift the elements.

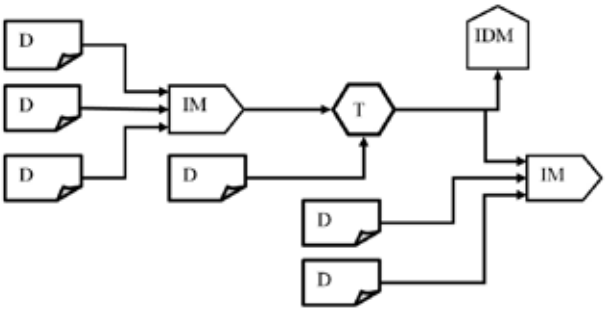


Fig 1 Example of MID

Discussion of Deliverables

Fig 1 gives a simple example of an MID.

On the left it depicts **Deliverables**. Each Deliverable has its own identification, from which you can derive:

- What is the content (Functionality, Interfaces)
- Which development group is responsible for creating it
- What level of quality does it have on delivery

The Deliverable describes a product in a certain, not necessarily completed, state. However, the stakeholders know what state it has.

In Fig 1 each 'document' symbol with a 'D' indicates the Deliverable. In reality you may add some references, e.g. to an Excel sheet giving details.

Each Deliverable may have its own 'sub' MID, which charts how the team develops this Deliverable. In this way the development teams can use the MID in a scalable way, and

ISSECO®
SECURE SOFTWARE ENGINEERING

ISSECO® – INTERNATIONAL CERTIFIED PROFESSIONAL FOR SECURE SOFTWARE ENGINEERING

In times of daily internet attacks, customers have grown accustomed to relying heavily on their software security. Every security breach can lead to a damaged reputation which in turn can result in customer loss. The development of secure software requires special knowledge of the entire software development process. Together with some of the best names in the field, iSQI has developed the Certified Professional for Secure Software Engineering.

CONTENT:

The Certified Professional for Secure Software Engineering guarantees knowledge about aspects of security in all phases of software development (software lifecycle):

· Requirements Engineering
· Design & Specification
· Programming & Testing
· Evaluation, Distribution & Maintenance

TARGET GROUP:

· Software Developers & Development Managers, Requirement Engineers, Quality Managers, Security Managers

CERTIFICATION:

· Certificate "Certified Professional for Secure Software Engineering"

FOR MORE INFORMATION VISIT: WWW.ISSECO.ORG

MEET US AT CONQUEST 2008 / www.conquest-conference.org
The conference program is included in this issue of testing experience.

iSQI International Software Quality Institute

show only the details that interest the rest of the team.

Discussion of Integration Moments

Fig 1 depicts three **Deliverables** that have some technically or logically necessary reason for integrating them together in the **Integration Moment**. This means that if one of the teams does not deliver any of the Deliverables, the **Integration Moment** cannot take place. The combination simply will not work. The Integration Manager (new project role, see: **Applying the method**) only groups elements together that have this relationship.
Each Integration Moment has its own identification which describes in detail what activities are needed to integrate the Deliverables. As an Example: The three Deliverables may stand for a motor movement control involving a:

- Motor
- Electronic motor controller/driver
- A software layer that generates movement profiles

Obviously, I could sum up a very large number of examples.

Integrating means:

- Generating the software
- Doing build tests on the software
- Connecting the electronics
- Showing 'basic functionality'
- And so on, depending on your organization

Discussion of Test Moment

In order to test, the tester needs access to the delivered product and he must also have the test environment up and running. Most importantly, he must know the purpose of this test. In our example that would mean:

- Test documentation
- Supplying a (laboratory) power supply
- Organizing a local PC connected to the electronics
- Designed test scripts
- Creating and downloading test scripts
- Having a means of verifying that the motor makes the expected movements, has vibration within the accepted range and any other measurement equipment needed to test any parameter the project should control at this time
- And some other standard tools like reporting and so on

The Deliverable leading straight into the Test Moment symbolizes all equipment and other test environment needs that a team must specifically develop or buy. Naturally, the Test Moment also has a document it refers to which describes the test purpose, the types of test done, acceptance criteria and so on.

Discussion of Increment Delivery Moment

This symbol indicates when the teams have developed sufficient functionality to deliver this to a customer. This may include a formal validation, and possibly acceptance tests by the customer, and handing over of user documentation as defined by your organization. Teams may also use this symbol to indicate a delivery to an MID at a 'higher level'. In that case, other 'light-weight' acceptance criteria may apply. A product may have several of these moments, enabling evolutionary development.

Applying the method

Unless the relevant stakeholders understand the MID and wholeheartedly support it, it will not do much good and just gather dust in some drawer. We therefore introduce the role of Integration Manager. A designer or senior tester may fulfill this role. As a first step he makes an overview of the stakeholders for this project. He may then create an overview of what the integration activity will look like. Then he will organize 'beamer sessions' with the stakeholders (or a subgroup, depending on the exact content of the discussion) to fill in the details of the diagram. In a beamer session a secretary uses the PC to display the drawing, and to quickly make changes as proposed and agreed by the team. You may use a whiteboard for this. The Integration Manager makes certain that all persons present understand the technical issues, and the added value of testing at a certain moment. He also makes sure that designers can design and develop (or specify) the relevant test equipment.

Critical success factors

The method will only work if the project adheres to a number of critical success factors. These are:

- Stakeholders understand the way of working with the MID and support it
- Involvement of all stakeholders
- Minimum level of abstraction at which testers can partake in the discussion
- The motivation to see development of test tooling as real product development
- Cooperation across disciplines
- Good insight into what you will develop

Practical results

Within Philips Healthcare we have used the method several times.
Without exception, we found this leads to much better insight for stakeholders to understand what they contribute to the integration process. The project manager had better control, and projects could be completed faster. Also, due to faults that the testers found at an early stage, developers did considerably less rework than in earlier comparable projects, which is what led to faster project completion. Furthermore the MID helped manage unexpected behaviors in the outside world like a change in requirements or unexpected technical problems. In the latter case, the project team could easily identify which technical areas the technical problem affected.

Conclusions

Good use of the MID contributes to better project control and faster project throughput, with inherently better quality of the final product. Most people do not find it difficult to learn, and do not need additional tooling. In practice you can extend the ideas shown above by using colors or other indicators, in order to indicate information relevant to your organization. I have not discussed the relationship between MID and Configuration Management.

References

Dieter Arnouts: Test management in different Software development life cycle models, Testing Experience, June, 2008

Timo Käkölä; Juan Carlos Dueñas Lopez, Research Issues in Engineering and Management: E.S. Engelsma: Chapter 12: Incremental Systems Integration within Multi-disciplinary Product Line Engineering using Configuration Item Evolution Diagrams

Biography

The second Dutchman to pass the ISEB Practitioner level with 'distinction', he now works as Test Architect for Philips Healthcare. He has worked with ITEA projects doing research into product family development related test issues, and has given several presentations at SQS congresses and other organizations. In the past he was also employed as system designer for MRI equipment, and was head of a testing department for 8 years during which he introduced TMM and started a group building Automated Test Equipment. He spends most of his spare time studying and playing the clarinet, and performing with his Klezmer band at weddings and festivals, or he studies something else. Those 'in the know' about NLP-related aspects of documenting will notice NLP's influence on his article.

Diffusion Methodology for Regression Test Analysis

by Oron Dilmoni

Abstract

Regression tests were always a very hard task.

The difficulties stem from the fact that you touch the code and you should select the relevant test cases for retesting.

But, what are those test cases? How can you decide which test case to retest and which one to skip? How can you be sure that you analyze the system correctly and select all the test cases that were affected by the changes? And what happens if there were several changes?

IEEE defines regression testing as *"...the way to catch a large class of these bugs quickly and efficiently.* ***Regression testing focuses on ensuring that (theoretically) everything that used to work still works.****"*

But how can you do it quickly and efficiently?

The answer, surely, must be based on "Risk Analysis", but we must remember that the reasons for retesting are the changes in the code. So the most risky issue should be the changes, and accordingly you should manage all your risk analysis based on this.

If you are a global company dealing with rollouts, then your problem is much more complicated .The SW is developed for one country or branch and then it is being customized and propagated to another country or branch. However, by customizing the code for the second country or branch (the first rollout), what happened to the first country? What tests should I do in order to ensure that it will not be impacted? And what will happen in the second, third (and so on) rollouts?

This article describes a new methodology named "The Diffusion Methodology for Regression Test Analysis", which enables you to manage a quick and efficient regression test mechanism.

The methodology is meant for organizations which have a lot of regression test cycles, especially for global companies dealing with rollouts.

Introduction

Regression tests are a very important part on our lives as SW testers.

We come across them almost everywhere and all the time: after bug fixing, after change requests have been implemented, after version upgrades, etc.

The first question we should ask is: "When" should we perform regression tests?

As a matter of fact, surveys show that only about 30% of software development relates to new developments, and the rest are maintenance, support, bug fixing, version upgrades, etc. Therefore, our main work as SW testers is to test changes in existing systems, which means regression tests.

There are a lot of definitions for the term itself, but there is no doubt that it basically deals with changes in the code.

IEEE defines the term as follows: *"Many products go through several releases, and can be ported to many platforms (hardware and/or operating system). Every line of code written or modified offers opportunity for bugs to creep in. Regression testing is the way to catch a large class of these bugs quickly and efficiently. Regression testing focuses on ensuring that (theoretically) everything that used to work still works."*

The FDA GPOSV (January 11,2002) definition reads: *"Regression testing is the rerunning of test cases that a program has previously executed correctly and comparing the current result to the previous result in order to detect unintended effects of a software change."*

Therefore, we can absolutely conclude that regression tests should be done after every change in the code. (As we will see later, this doesn't mean we should test the whole system. The term 'regression test' refers to the overall process as it will be detailed in this article and the methodology which offers a mechanism to analyze the change and decide about the next steps).

Note: It should be noted that the proposed methodology is a very useful tool. The way to use this tool (including the scales, formulas, risk analysis, etc. proposed here) might be changed in order to suit the specified organization.

Regression Test Flow & Stages

In principle, as mentioned in the abstract, when inside the project, we must make decisions regarding the regression tests quickly. Therefore, we do not want them to be taken randomly, by the wrong persons, on the basis of wrong circumstances or missing information. So the idea is to create a consistent analysis method for all code changes.

The flow is divided to 4 stages:

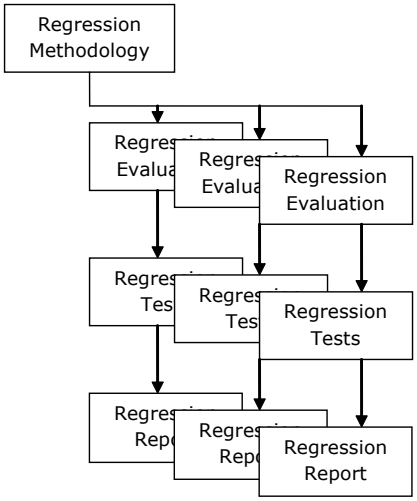


Figure 1 – Regression Tests Basic Flow & Stages

Stage 1: Regression Methodology – Each organization shall once create its own methodology regarding the regression tests, based on regulations and the company policy. The methodology shall define the whole process, responsibilities, and milestones, and it shall describe how we are going to handle each type of "When"?

Stage 2: Regression Evaluation - Analyzing the impact of each change (when it occurs) based on the review of the relevant documentation and in accordance with the specific strategy 'event'.

Stage 3: Regression Tests - Regression tests should be performed and should be focused on several aspects: new processes, changed processes, deleted processes, affected processes and bug fixing.

Stage 4: Regression Report – Summarizing the results and deciding about more cycles as deemed necessary.

Stage 1 - Regression Methodology

Each organization or sometimes each project, shall define how it is going to deal with the Regression tests issue. The organization shall adopt a set of rules which will later help to take the decisions quickly. This article might, for example, be adopted as a methodology, but even if this is the case, the organization shall still consider specific issues which are only relevant to it.
When building the Regression methodology, you should consider at least four important issues:

1. Risk Analysis methodology for Regression tests.
2. Project Influence Map.
3. Cause & Results plan.
4. Responsibilities.

Risk Analysis for Regression
While thinking on the first issue, we can determine that Risk Analysis will definitely help the organization to identify the processes which were not changed at all and those which were not even impacted.
Let us agree that:

Risk = Severity * Probability

Then under the assumption that the tests will cover all the system and the system worked fine before the changes, the probability for bugs in processes which were not changed is:

Probability -> 0

So we can conclude that:

Risk = Severity * 0
=> Risk = 0

And therefore, we can define in our Regression methodology that **the severity should be considered only if the process is affected**.

The Regression Strategy is based on Risk Analysis. The risk analysis might consider any risk types (finance, business, regulations, etc.) but just like when we define regression tests, it should be focused on the processes potentially affected by the change.
The Risk Analysis should first find the affected processes and focus on these. The other types of risk are not relevant if the process was not affected (they might be relevant if the process is affected). The other 'high-risk types' might be integrated in the methodology and re-tested as a company decision but not enforced.

So, how we will find the affected processes?

Building Project Influence Map
As the regression decision should be requirement-based and not testing-based, the idea is to already build an influence map in the specification phase. While defining all the business processes of the system and all the requirements, the system analysts should build a very simple map which will point out the reciprocation between each 2 processes. This pre-analysis is done by the professional analysis team at the right stage and, apart from saving time later on, it helps the project team to working consistently. This is because the whole team will work with the map and decisions will not be taken randomly by the first person to get it.

This might be done by using the following simple table:

Process (Req.)	Affected processes
1. Order to Cash	2
2. Inventory Management	---
3. Turning on the system	---

Figure 2 – Project's Processes influence Map

In the Process column we should map all the processes from the process tree. In the "Affected processes" column, the team should analyze only the processes which will be affected **directly** by any change in the process. (As we are working with a diffusion method, it is very important to map only the direct influence).

Cause & Result plan
As part of the Regression Methodology, the project management should think about as many situations of unexpected changes as possible, and define what the required actions for each situation are.

Responsibilities
As part of the Regression Methodology, the project management should identify the exact persons who will have the right to perform or to approve each operation. For example, who will build the influence map, who will approve changes in that map, who is going to approve bug fixing, who is going to analyze the required regression tests for that change, and so on.

Stage 2 - Regression Evaluation

Once we have the Regression methodology, the infrastructure is ready and we can handle each change in the project and decide about the required regressions efficiency and quickly. At the regression evaluation stage, the unaffected processes are basically sifted out, and the other affected processes which are likely to be risky are grouped.

This is done in 3 stages:

1. Building the Regression Influence Map
2. Recursive Diffusion (3 cycles)
3. Filtration (after deciding about the regression threshold)

Building Regression Influence Map
At this stage, we have changes that were done to the system, and we have the influence map which was designed in the specification stage. These should now be combined into one table.

This could be done using the following simple table:

Process (Req.)	RCat	Affected processes	RFactor
1. Order to Cash	3	2	60
2. Inventory Management	4	---	80
3. Turning on the system	5	---	100

Figure 2 – Project's Processes influence Map

The Process column and the Affected Processes column already exist. We only have to evaluate the changes and fill in the RCat & RFactor columns.
The RCat is the (Regression Category), which is represented by one of 3 options (3 - Processes that were not changed since the last tests; 4 - Processes that were configured and changed by system configuration mechanisms; 5 - Processes which were changed and involved development/changes in the code.):

RCat = 3 or **4** or **5**

And the RFactor:

RFactor = 20 * RCat

At that point, we can already easily find the processes which were directly affected by sorting the influence table.
In fact, all processes marked with RFactor = 100 are risky/ affected and must be retested. All processes marked with RFactor = 60 are not directly affected.

Recursive Diffusion
In the Diffusion stage, the objective is to find the processes that might be affected transitively (not directly). For example, if process A affects directly process B and process B affects directly process C, then process C might be affected by any changes in process A transitively. The depth of the transitive relation might be controlled by the organization according to the number of recursive evaluations performed.
The recursive evaluation is done only to processes classified as Rcat 4 or 5, because pro-

Díaz Hilterscheid

© iStockphoto

Wachsen Sie mit uns!

Die Díaz & Hilterscheid Unternehmensberatung GmbH (www.diazhilterscheid.de) unterstützt seit 1998 individuell in den Beratungsfeldern Financial Services, IT Management & Quality Services, Training Services und IT Law.

Die Mehrzahl unserer Kunden sind Kreditinstitute, Versicherungen und IT-Dienstleister für Finanzinstitute. Im Zuge unseres weiteren Wachstums suchen wir zur Verstärkung unseres Bereichs IT Management & Quality Services zum nächstmöglichen Termin mehrere:

Senior Consultants IT Management & Quality Services (m/w) für unsere Kunden aus der Finanzwirtschaft

Deutschland
Europa

Ihr Profil:

- Fundierte Ausbildung oder Studium (Informatik, BWL, Mathematik, etc.) sowie mehrjährige Berufserfahrung als IT-Consultant in einem Fachbereich bzw. in der Organisation eines Finanzunternehmens, eines finanzwirtschaftlichen IT-Dienstleisters oder in einem Beratungsunternehmen in entsprechenden Projekten
- Erfahrung in der Führung von großen Teams (Projektleiter, Teilprojektleiter, Testmanager)
- Kenntnis und praktische Anwendung von Methoden und Standards, wie CMMI®, SPICE, ITIL®, TPI®, IEEE, ISO 9126
- Eigeninitiative, Verhandlungs- und Gestaltungsgeschick, repräsentatives Auftreten
- Hohe Reisebereitschaft
- Gute Englischkenntnisse in Wort und Schrift

Wir bieten:

- Interessante und anspruchsvolle IT-Projekte in den Bereichen IT-Prozesse, MA-Qualifikation und Coaching, Softwaretest sowie Toolauswahl und -einführung
- Weiterentwicklung in einem flexiblen Unternehmen mit eigenen Gestaltungsspielräumen
- Attraktives, erfolgsbezogenes Gehalt

Ihr Kontakt:

Bitte senden Sie Ihre aussagekräftige Online-Bewerbung an unseren Bereich Personal (hr@diazhilterscheid.de). Ihre Fragen im Vorfeld beantworten wir gerne (+49 (0)30 74 76 28 0).

cesses classified as 3 are not affecting anything (3 = processes that were not changed). It is recommended that the phase is performed in another 2 recursive cycles according to the following formulas:

- For cycle II:

  RFactor' = RFactor' + RFactor/10

Where RFactor' is the RFactor of any process in the 'Affected Processes' column of the original process.

- For cycle III:

  RFactor'' = RFactor'' + RFactor'/100

Where RFactor'' is the RFactor of any process in the 'Affected Processes' column of the 2nd level of affected processes.

Note: In very solid organization, it is possible to perform more and more recursions. The formula in each cycle shall be changed only by another division by 10, because the level of influence becomes more and more indirect and therefore the risk reduces accordingly.

Filtration

At this stage, we already have an updated influence map, where the RFactor has actually become the decisive criterion. The higher the RFactor, the higher the risk that the process is affected by the changes.

We might find a lot of RFactor values for the different processes in the table. All we have to do is to define the RFactor threshold and to filter the table accordingly – **only processes whose RFactor is higher than the threshold should be retested.**

The role of the filtration process is to filter all the unaffected processes.

As stated in the introduction, unaffected processes should not be retested (no matter what the severity is):

If **RFactor = 60 => No regression on this process**

And since all processes which might be directly affected must be retested again,

we have to identify the highly risky processes which might be directly affected or which have a lot of 2nd level impact:

If **RFactor > 100 => Regression must be executed**

All processes which are between 60 and 100 will be filtered by the 'Regression Threshold'. This threshold might be decided based on the existing RA, considering also the severity of the process (requirement).

If **RFactor > 'Regression Threshold' => Regression must be executed**

If **RFactor < 'Regression Threshold' => No regression on this process**

The Regression Threshold should be decided by the organization and might be changed according to different needs. In principle, all the processes which are not affected are allocated an RFactor of 60. Each "affection" of the first recursive cycle can contribute "8" or "10" to the RFactor value of the affected process, and each affection of the second recursive cycle can contribute "0.8" or "1.0" to the RFactor value of the affected process.

This means that if RFactor is higher than 80, the process might be changed directly by customization, or it is at least 2-3 times directly affected, or it has more than 20 2nd level affections. So, it will be good to decide that:

Regression Threshold < 80

However, if we analyze the RFactor further, we will find that even if the RFactor = 68, it means that the process is directly affected, or it has more than 10 2nd level affections.

So, it will be good to decide that:

Regression Threshold < 68

Therefore, it is recommended to define the Regression Threshold as:

Regression Threshold = 68

(But in any case it will be: 60 < Regression Threshold < 68)

Let's take the following example:

Process (Req.)	RCat	Affected processes	RFactor
A			60
B			60.8
C			61
D			61.6
E			61.8
F			62
G			62.4
H			62.8
I			64
J			68
K			70
L			76
M			80
N			88
O			90
P			96
Q			98
R			100
S			106

Figure 4 – Example: Regression influence map

- Process A – was not affected at all by any change – will not be retested.
- Process B – was affected only once in 2nd level from one customization – will not be retested.
- Process H – was affected only in 2nd level (2 times by changes in the code and one time from customization) – will not be retested.
- Process I – was affected only in 2nd level (4 times by changes in the code, or 5 times by customizations) – will not be retested.
- Process J – was affected directly by customization or even if not directly, then it was affected at least 8 times by 2nd level – will be retested.

Note: Even though the process looks complicated, the effort might be assimilated very easily and its contribution will have an impact very soon.

Stage 3 - Regression Tests

When working with a strong methodology, the Regression Tests themselves become the very easy part. At this stage, you should only execute the selected test cases according to the tables from previous stages.

Of course, it is necessary to track the execution and to identify whether there are any areas with a lot of problems. If we identify, for example, a process which has a lot of failures, it is recommended to go back to the influence table and to re- evaluate the affected processes of this process.

In some cases, the above methodology might prove to be a much stronger tool, since you can perform the diffusion only on the identified failure process and also identify precisely the affected processes at all levels. In this case, it would be recommended to reduce the Regression Threshold for that evaluation and execute more tests.

Stage 4 - Regression Report

Like every other test, the regression cycle should be summarized and reported (by producing a test report). The more important role at this stage of dealing with regression is to decide whether more rigorous operations are required.

We should remember that sometimes processes are being added, and they should also be added to the influence map with all their relations. Likewise, sometimes processes are deleted, and they should also be removed from the other process relations. In addition, as processes are updated, they might affect more processes or stopping affecting others.

So, after finalizing the cycle, it is necessary to go back to the influence map and check if it needs to be updated.

Advantages

The advantage of this methodology is first and foremost to have a methodology for Regression Tests, when usually organizations do not function in this field with an approved methodology, but simply let projects flow.

Apart from this, there are more specific advantages:

- All the changes are handled by the same mechanism, no matter if it is a major version upgrade, minor version, change request, bug fixing, or extensions and so on.
- The thinking part is done earlier (during the specification), which will save time during later stages, which are much more critical for the project.
- This thinking is done consistently by the correct persons and not randomly by the wrong persons.
- The methodology enforces good practices and correct procedures, so the projects become well documented and structured.
- The risks are reduced through mapping the affected processes exactly, so there

are much fewer cases of failure after regressions.

- The scope of the regression tests might be calculated much earlier (while planning the changes), and it is possible to create very accurate plans.
- The methodology helps to get the correct decisions for unexpected occurrences during the project life cycle (as the map tool might be used for reactions to failures detected and for re-planning), so the organization becomes much more flexible for changes.
- The methodology enables a unique language and unique procedures within the organization, since all projects and all team members work with the same methods and with the same actions.

Summary

see Fig. 5

References

[1] – FDA General Principles for Software Validation (January 11,2002).
[2] - GAMP Good Practice guide: Testing of GxP systems (2005)
[3] - GAMP 4 "Guide for Validation of Automated Systems" –issued in December 2001
[4] - IEEE STD 829-1998, "Software Test Documentation" – updated in 1998.

Biography

Oron Dilmoni has been working in the software testing area for 10 years. Oron started is career as a SW testing engineer and during the years he became a Testing Project Manager, Senior Testing Project Manager and Practice Manager. During his career, Oron managed dozens of SW testing projects in various fields, sizes and types. In his latest role, Oron was a branch manager at Tescom Israel. Oron established a nearshoring branch supplying SW testing services for the Israeli market.

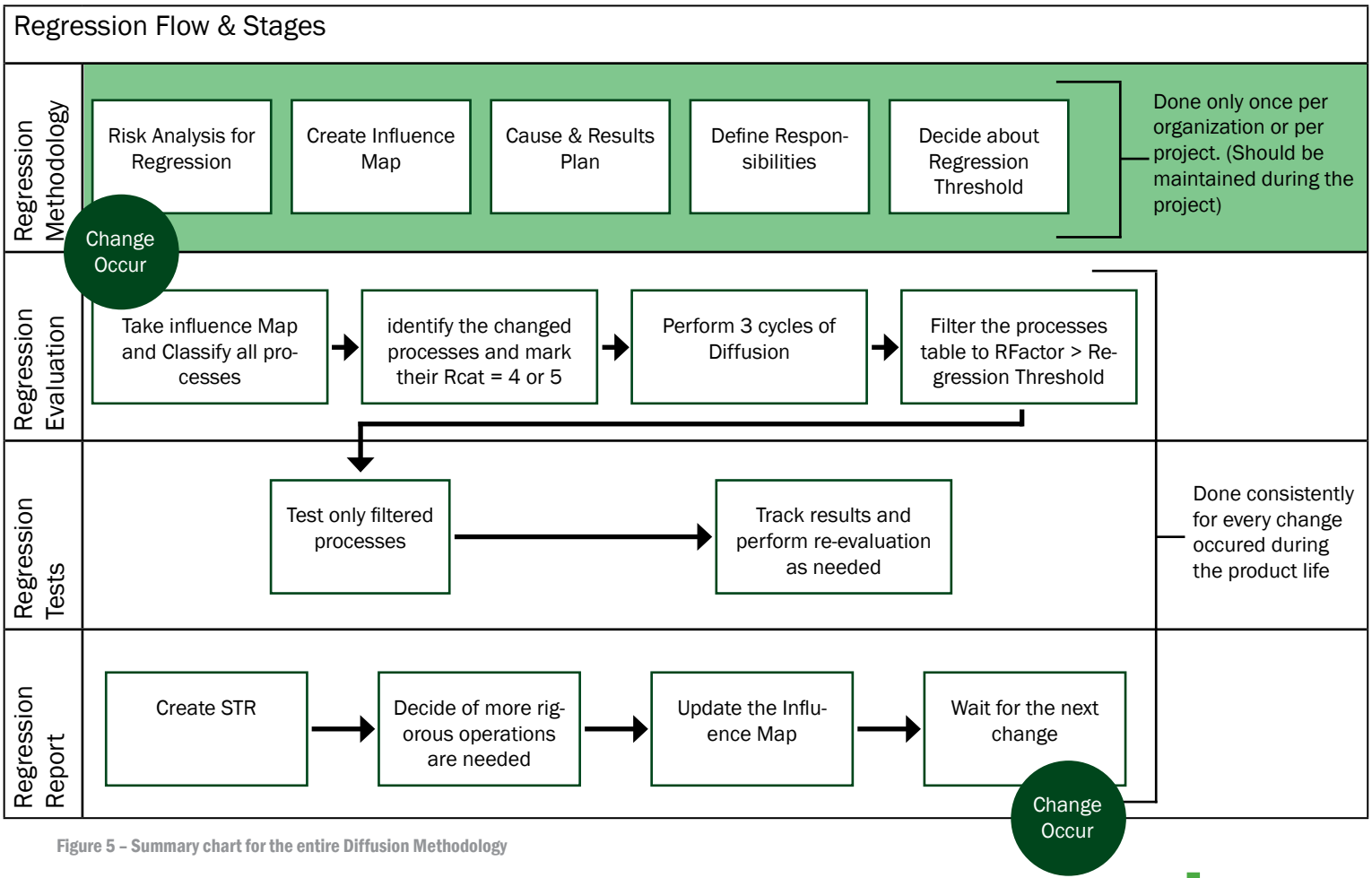
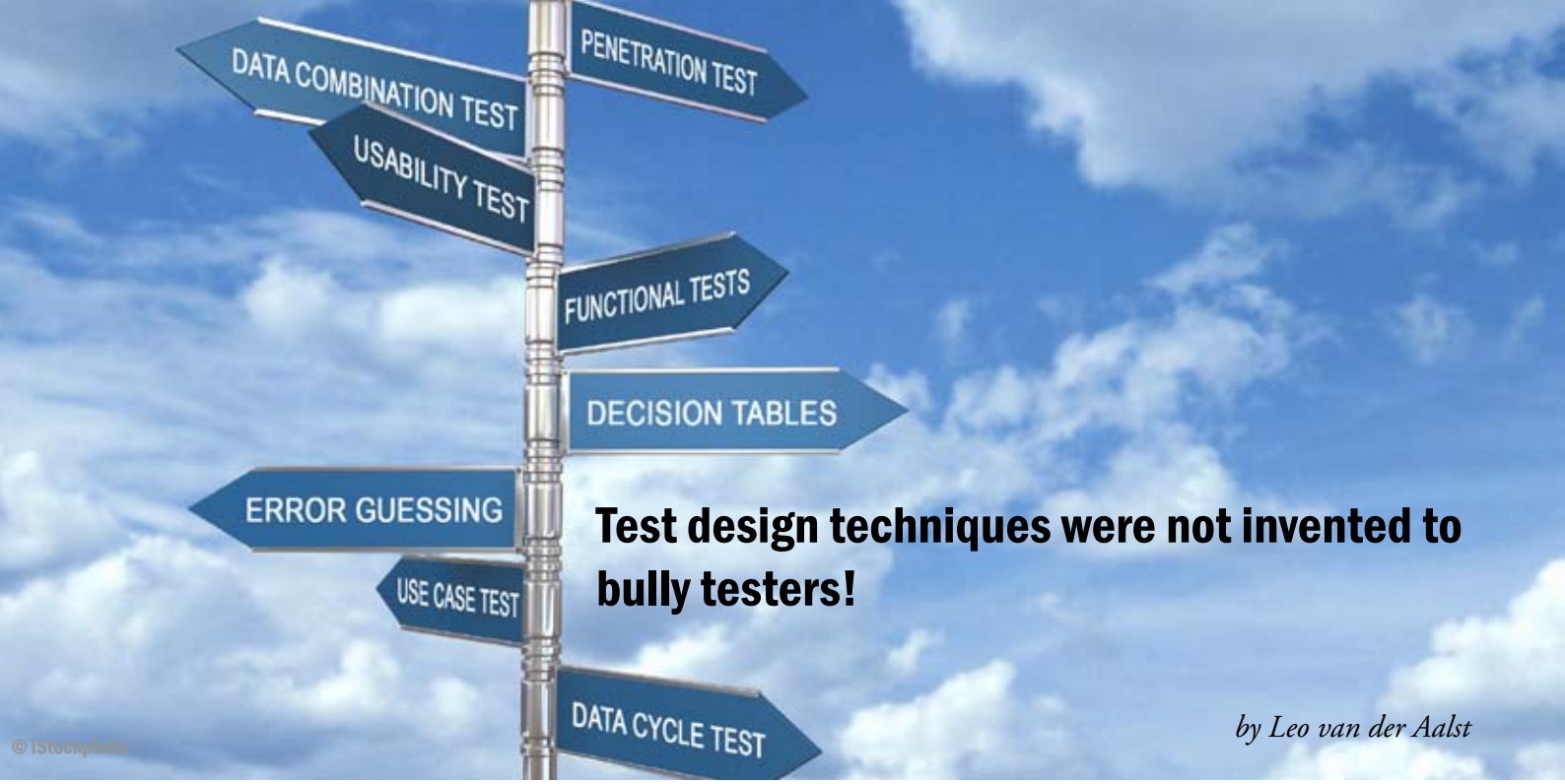


Figure 5 – Summary chart for the entire Diffusion Methodology



Test design techniques were not invented to bully testers!

by Leo van der Aalst

Frequently, clients and even testers complain that using test design techniques is a difficult and time-consuming business. If they can get away with it, they would prefer not using any techniques at all! That's a pity, because these techniques represent the only way to realise the agreed test strategy in a *demonstrable* way. This article provides you with the tools to select one or more suitable techniques.

Substantiate the test strategy with test design techniques

After the test goals and product risks have been established via a product risk analysis, the resulting test strategy should be substantiated, according to the intensity of testing for a specific combination of characteristics and object parts.

IN MORE DETAIL

- A test goal is a success criterion for the test assignment specified in the customer's language.
- A product risk is the chance that the product fails in relation to the expected damage if it does so.
- The test strategy is the distribution of the test effort and test intensity over the combinations of characteristics and object parts aimed at finding the most important defects as early as possible and at the lowest costs.
- The intensity of testing is light, average or thorough; it is part of the test strategy.
- Characteristics include amongst others functionality, user-friendliness and security.
- Object parts are usually the sub-systems of the application software.

Having determined the characteristic to be tested and the test intensity, one or more suitable test design techniques can be selected to create the test cases (See Figure 1: From test goals to test cases, below).

Figure 1: From test goals to test cases

TMap Next [Koomen et al, 2006] describes a large number of techniques and a great many other techniques can be found in books and on the Internet – including some created by testers themselves. Of course, variations of any technique can be also generated.

If an unsuitable technique is selected or no techniques are used, all of the previous steps will have been in vain, and it becomes very difficult to make a judgement as to whether the test goals have or have not been realised – with the attendant risks. For instance, the chance of production disruptions grows - unfortunately still a common situation. And in terms of test governance, test goals cannot be traced through to test cases.

Are test design techniques difficult and time-consuming?

Why is it that clients and testers often consider that using test design techniques is a difficult and time-consuming business? It would seem this is a matter of 'unknown, unloved'!

The quantity and abstract nature of techniques is an important cause of their being 'unloved'.

To avoid overloading the tester such that he 'can't see the wood for the trees', it is better not to teach or explain every single test design technique. Practice has shown that if the techniques are explained well, with examples from a tester's own immediate work environment, the tester 'suddenly' does not feel they are so difficult and sees the benefits of using them. The reason for this is that using examples from the tester's work environment eliminates the abstraction of a technique, and allows the tester to see its practical application and motivates him to put into practice at his workplace what he has learned.

Also some organisations want more 'certainty' and assurance that their testers have an adequate knowledge of test design techniques. Solutions include encouraging or even mandating their in-house testers to acquire formal certification or asking for external certified testers when testing.

Often, when the use of test design techniques is described as time-consuming, people forget that the techniques may be used 'incorrectly'. The aim is not for the tester to design every possible test case, but rather that he selects a specific technique in relation to the selected test strategy - aiming to achieve the highest possible 'defect-finding chance' with the least possible number of test cases. In practice, we find that testers frequently make the wrong choice. As a result, an excessive number of test cases are designed. This is an important cause for considering the use of techniques to be time-consuming.

IN MORE DETAIL

Defect-finding chance
Let's say that you have a 'travel reservation system' with the following parameters and equivalence classes:
Number of days : 8; 15; >15
Amount (euros) : <500; 500-1000; >1000
Membership card: none; silver; gold; platinum
Departure date : workday; weekend; bank holiday
You need 3x3x4x3=108 test cases to test all possible combinations (the complete decision table). If you use the technique 'pairwise testing', you only need 13 test cases (if using the 'Allpairs' tool[1]).

Research conducted by the National Institute of Standards and Technology [Kuhn, 2000] shows that 98% of all defects are found when pairwise testing is used. This is because just 2% of all defects are caused by a problem in the combination of three parameters or more! In other words, 12% of all possible test cases will be enough to detect 98% of all defects in the above example.

Choosing the best possible technique

After the test strategy is determined, suitable techniques must be chosen. This is not always easy - after all, we must take a large number of variables into account:

- characteristic
- test intensity
- test basis
- knowledge and skills of the testers
- labour-intensiveness of the technique.

The flow chart illustrated in Figure 2: Technique selection diagram is a valuable means of making the technique selection, in conjunction with the information in Table 1: Proposed technique for a specific combination of characteristic and selected test intensity, and Table 2: The required test base for a proposed technique.

In Figure 2, the activity 'Take measures' may include:

- Required test basis not available
  - ask designers to adapt the test basis so that the technique can be used
  - ask testers to adapt the test basis so that the technique can be used
  - organise information sessions to achieve a usable test basis.

- Inadequate knowledge and skills on the tester's part
  - train the testers in the proposed technique
  - select another technique because it is a better match with the tester's knowledge and skills.

- Labour-intensiveness disproportionate to the time available
  - make the test less intensive
  - make more time available.

Clearly, the proposed measures must be agreed with the client, since they may have an impact on the agreed result, the risks to be covered, the estimated costs, and/or the planned time.

The technique that is the result of using this method of selection is a suggestion only; there may, of course, be reasons for selecting another technique. The selection diagram is just a tool. Finally, sometimes, a certain technique is imposed on a tester, for reasons of industry regulation or standardisation.

IN MORE DETAIL

Technique is imposed
Not every company, industry or specific application allows a tester to 'freely' select a technique; it may be prescribed. An example from the aviation industry illustrates this situation. Aircraft can only use software that aviation authorities have found to be "safe" for aviation purposes. To this end, the American Radio Technical Commission for Aeronautics (RTCA) and the European Organization for Civil Aviation Equipment (Eurocae) have developed a standard: DO-178B for America and ED-12B for Europe. These standards classify software systems according to the consequences for the aircraft and its passengers if a system should fail. The consequences range from 'no negative impact' (level E) to 'catastrophic' (level A). DO-178B and ED-12B require the coverage type "decision points modified condition/ decision coverage (MCDC)" for level A testing. The American and European aviation authorities, the Federal Aviation Administration (FAA) and the European Aviation Safety Agency (EASA), accept this standard for certifying aviation software systems.

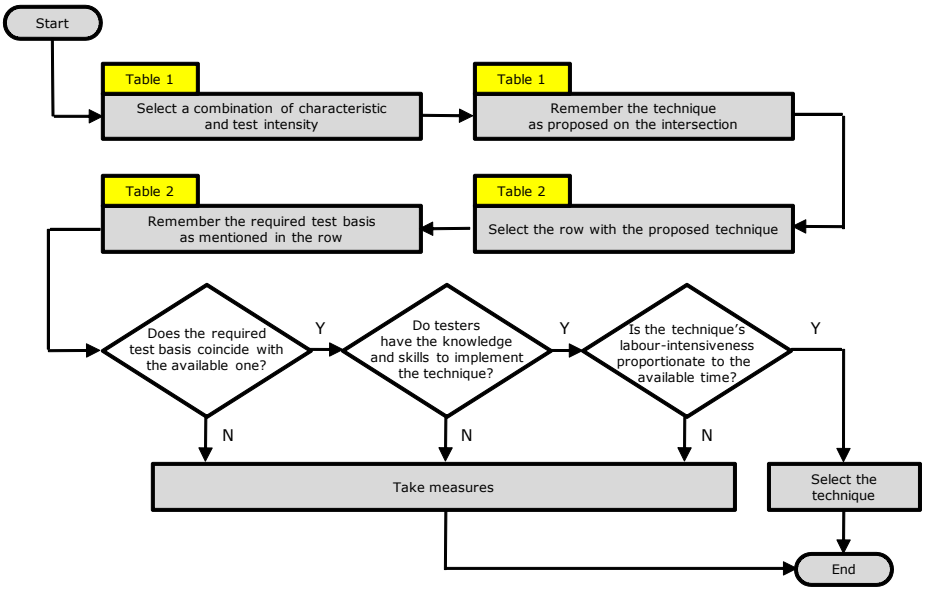


Figure 2: Technique selection diagram

[1] The 'Allpairs' tool was created by James Bach and can be downloaded from http://www.satisfice.com. Another tool is 'Pict33' by Microsoft®, which can be downloaded from http://www.pairwise.org. The DaimlerChrysler tool, 'Classification tree editor', can also be used; this can be downloaded from http://www.systematic-testing.com.

Characteristic	Test intensity		
	Light coverage	Average coverage	Thorough coverage
Manageability	Checklist PCT-test depth level 1 UCT-checklist EG	DCoT-equivalence classes PCT-test depth level 2 ET	DCoT-pairwise testing PCT-test depth level 3
Security	Checklist EG	DCoT-equivalence classes SEM-modified condition/decision coverage ET	DCoT-pairwise testing Penetration test
Usability	UCT-checklist EG	PCT-test depth level 2 UCT-paths	RLT-operational/load profiles UCT-decision points
Continuity	EG	RLT-operational/load profiles ET	RLT-operational/load profiles
Functionality - detail	DTT- condition/decision coverage DCoT-equivalence classes ECT- condition/decision coverage EG	DCoT-pairwise testing ECT-modified condition/decision coverage ET	DTT- multiple condition coverage (+ boundary values) DCoT-N-wise testing ECT-multiple condition coverage
Functionality - overall	DCoT-equivalence classes SYN-checklist (limited) UCT-checklist EG	DCoT pairwise testing DCyT (life cycle of the data) CRUD DCyT (integrity rules) decision coverage PCT-test depth level 2 SYN (prioritised list) SEM-condition/decision coverage UCT-paths ET	DCoT-N-wise testing DCyT (life cycle of the data) CRUD (extra Rs) DCyT (integrity rules) modified condition/decision coverage RLT-operational/load profiles SEM-modified condition/decision coverage UCT-decision points
Functionality - validations	SYN-checklist (limited) EG	SEM-condition/decision coverage SYN (prioritised list)	SEM-modified condition/decision coverage
User-friendliness	SYN-checklist (limited) EG	PCT-test depth level 2 SYN (prioritised list) UCT-checklist	Usability test (possibly in lab)
Infrastructure (suitability for)	EG	RLT-operational/load profiles ET	RLT-operational/load profiles
Suitability	UCT-checklist DCoT-equivalence classes PCT-test depth level 1 UCT-checklist EG	UCT-paths PCT-test depth level 2 DCoT-pairwise testing DCyT (life cycle of the data) CRUD DCyT (integrity rules) decision coverage ET	RLT-operational/load profiles UCT-decision points DCoT-N-wise testing DCyT (life cycle of the data) CRUD (extra Rs) DCyT (integrity rules) modified condition/decision coverage PCT test depth level 3
Performance	EG	RLT-operational/load profiles ET	RLT-operational/load profiles
Portability	Checklist Random sample functional tests Random sample environment combinations EG	Functional regression test Important environment combinations ET	All functional tests All environment combinations
Efficiency	EG	RLT-operational/load profiles ET	RLT-operational/load profiles

Table 1: Proposed technique for a specific combination of characteristics and selected test intensity

Please refer to Table 2 for the meaning of the abbreviations. See TMap Next [Koomen et al, 2006] for comments on the test design techniques.

Comments on the terms used in Table 1:
Portability - functional tests
When testing portability, a random sample of functional tests, the regression tests or all test cases can be executed in a specific environment with increasing test intensity.
Environment combinations
Testing portability determines whether the system runs in various environments. Environments may consist of different parts, such as hardware platform, database system, network, browser and operating system. If the system needs to be able to run on 3 (versions of) operating systems under 4 browsers (or browser versions), you already have 3 x 4 = 12 environment combinations to test.
Penetration test
The penetration test aims to find gaps in the system's security. It is usually executed by a so-called 'ethical hacker'.
Usability test
A test in which the users simulate business processes and test the system. Statements about the test object's user-friendliness are made by observing the users during the test. A specifically configured and controlled environment, which includes e.g. video cameras and a room with mirrored glazing for observers, is also called a usability lab.

Technique	Test basis								
	All types of test basis	Individual conditions or decision tables, without structure	Structured functional specification (pseudo code)	CRUD matrix, data integrity rules	Structured description of business or operating processes	Operational profiles, load profiles	Input and output specifications, business rules	Input and output specifications, attribute descriptions	Use cases
Checklist	x	x					x		x
decision table test (DTT)	x	x	x						
data combination test (DCoT)	x	x	x	x					
error guessing (EG)	x	x	x	x	x	x	x	x	x
exploratory testing (ET)	x		x			x	x		
elementary comparison test (ECT)		x	x						
functional tests		x					x		
Data cycle test (DCyT)		x		x					
Environment combinations		x					x		
Penetration test									x
process cycle test (PCT)		x		x	x	x		x	x
real life test (RLT)		x				x			x
semantic test (SEM)							x	x	x
syntactic test (SYN)							x	x	x
usability test								x	x
use case test (UCT)					x	x		x	x

Table 2: The required test base for a proposed technique

References

- [Koomen et al, 2006]
  Koomen, T., Aalst van der, L., Broekman, B., Vroon, M. (2006), TMap® Next for result-driven testing, Tutein Nolthenius, 's-Hertogenbosch, Netherlands, ISBN 90-72194-80-2, www.utn.nl
- [Kuhn, 2000]
  Kuhn, D.R., Wallace, D.R., (2000), Failure modes in medical device software: an analysis of 15 years of recall data, National Institute of Standards and Technology, Gaithersburg, MD 20899 USA, http://csrc.nist.gov/staff/Kuhn/final-rqse.pdf
- TMap® is a registered trademark of Sogeti Nederland B.V.

Biography

Leo van der Aalst has more than 20 years of testing experience and developed amongst others services for the implementation of test organisations and for test outsourcing.

He is co-author of TMap Next, designed the EXIN tasks for TMap Next certification and holds a readership position for quality&testing at the 'Fontys Hogescholen'.

Besides all this, Leo is a much sought-after teacher of test training, a regular speaker at national and international conferences, and he is the author of several articles.

Masthead

EDITOR
Díaz & Hilterscheid
Unternehmensberatung GmbH
Kurfürstendamm 179
10707 Berlin, Germany

Phone: +49 (0)30 74 76 28-0
Fax: +49 (0)30 74 76 28-99
E-Mail: info@diazhilterscheid.de

Díaz & Hilterscheid is a member of "Verband der Zeitschriftenverleger Berlin-Brandenburg e.V."

EDITORIAL
José Díaz

LAYOUT & DESIGN
Katrin Schülke

WEBSITE
www.testingexperience.com

ARTICLES & AUTHORS
editorial@testingexperience.com

PRINT RUN
12.000 printed issues
approx. 80.000 downloads

ADVERTISEMENTS
sales@testingexperience.com

SUBSCRIBE
www.testingexperience.com/subscribe.php

PRICE
free of charge

ISSN 1866-5705

The Test Maturity Model Integrated (TMMi ®)

Measuring our Capability to Deliver!

by Brian Wells

In a recent exercise I assessed the test processes of four CMMi Level 5-accredited organizations using one of the then available test maturity models. It may not be a surprise that all four organizations failed to achieve the same levels of capability for their test processes! However, while effective, the cosnsitency and effectiveness was open to question as, unlike Software Engineering generally, test does not have a uniformly accepted standard to measure itself against (and against others!).

I asked myself the question ***"If test disciplines have improved so much in recent decades, why do we not have the capability to accurately assess our capabilities (internally and against industry norms), realize our weaknesses and use this information to improve?"***.

It is because there is not an accepted ***Standard*** Reference Model within the public domain that is complete, fit for purpose, adaptable, useable, cost effective and able to satisfy the requirements of the industry for today and tomorrow.

In this article I will outline:

- The Battleground – Looking at the backdrop to the emerging awareness that we need to have the ability to assess the capabilities of test processes
- The origins of our species – Looking at the origins of Test Reference Models
- The emerging TMMi model – Looking at the TMMi® Test Reference Model – ***developed by test practitioners for all test practitioners!***

The Battleground!

It is true to say that Testing as a discipline has matured greatly over the last 2 decades and organizations are very actively looking to develop/implement/improve the capability of their test processes across the test and project life cycles and across the organization. All this against the back drop of the changing nature of testing and of Software Engineering overall.

It is also valid that Test activities (including all "forms" of testing; Reviews, Static, Dynamic Test activities) are the prime mechanism to demonstrate levels of required quality (however defined) – you could argue that Test is a Quality Assurance function, but that is another discussion!

Since the 1950's, Software Engineering has been a developing entity. In the beginning there was (arguably) chaos which led to the development of standard approaches for specific elements of the life cycle (often driven by individuals/organizations). Throughout the 1970's and 1980's these started to be published and accepted (i.e. the V Model). This standardization proceeded at pace and the industry identified a need to develop an overarching ability to evaluate the capability (or maturity) of the overall Software Engineering process; not just the individual elements.

In the 1990's this culminated in the emergence of standard reference models to measure capabilities of overall process maturity (CMM, CMMi, ITIL, ISO 9000, etc). These have developed into internationally accepted standards and public accreditations for organizations globally (although the original intention may have differed i.e. CMM was developed to measure the US Government contractors but was subsequently used to measure the process). These defined a framework of progression linked to capabilities and maturity, encompassing the developing "standard" ways of approaching elements. These are still evolving.

Testing as a separate, professional activity only really emerged in the 1990's but is developing the "hunger" for standards to evaluate their process much more quickly.

In this time, we have seen many developments in testing and software delivery; "fashion" times as well as very practices etc. We have also seen many changes in approaches; often driven by wider pressures such as economic globalization or customer expectations!. Throughout all this, the widely held view is that TESTING is now a major, publicly acknowledged professional activity (and profession?) within Software Engineering.

Given the proportion of IT spending on test activities, are these Software Engineering evaluation models (i.e. CMMi) sufficient to assess test activities in adequate depth? How mature and how professionally capable are the testing processes overall? How can this be demonstrated? Should we be concerned if we cannot accurately evaluate in a standard, transparently comparable manner?

Until recently, there was little for test to reference to provide robust standards and yardsticks to accurately measure our process capability by. Taking CMMi as an example, there are some elements that will evaluate testing, verification and validation, but this is only in the context of the Software Engineering process overall. Little attention is paid to evaluating in depth the capability of test and test-related quality processes and abilities.

The IT testing industry is developing standards for the process elements (i.e. ISTQB, BS 7925-2 etc). However, there is a real need

to supplement the existing Software Engineering capability models with one specifically targeted at Test providing us with the ability to evaluate the capabilities of a not inconsequential chunk of Software Engineering to a greater degree than hitherto we have done.

As a member of the Testing profession for over 2 decades I personally believe the profession needs a standard, internationally recognized reference model. It is only in this way that we, as a profession, can measure our capabilities. To do this, I also believe that this should be under the umbrella of an independent body providing an unbiased, independent role to the Test profession. Others think the same way!

The origin of the (TMMi®) Species!

There are a number of existing test "models" that have emerged in the past. These include, amongst the others:

- Boris Beizer's Test Model
- Gelperin and Hetzel's Evolution of Testing Model
- (elements of) Capability Maturity Model (CMM) & Capability |Maturity Model – integrated (CMMi)
- Test Process Improvement (TPI) model – Tim Koomen/Martin Pol
- Test Maturity Model (TMM) – Illinois Institute of Technology

While all are developing the ability to assess test process capability and indicating that the Test profession acknowledges the need they have limitations of one form or another. This has prevented "market penetration" and uptake as a developing de facto global standard.

Test practitioners, have used these models to try and evaluate weaknesses in test processes based on industry "knowledge" and extensive expertise and experience. However, while any offering had good elements, none fully satisfied the need for a comprehensive, understandable and flexible generic model that can adapt.

In 2004, a group of Test practitioners from across the world got together to discuss our experiences and the needs of the Test profession going forward. This group recognized the need for a standard test (process) capability reference model and, more importantly, an independent body to develop a standard offering. They agreed this should take the best of what is available and be enhanced based on the collective experience of the group and others.

Out of these discussions, they decided to create the TMMi Foundation as a non-profit making organization run by test professionals for the testing profession and beyond. The prime purpose was to create a standard Test Reference Model; TMMi®!

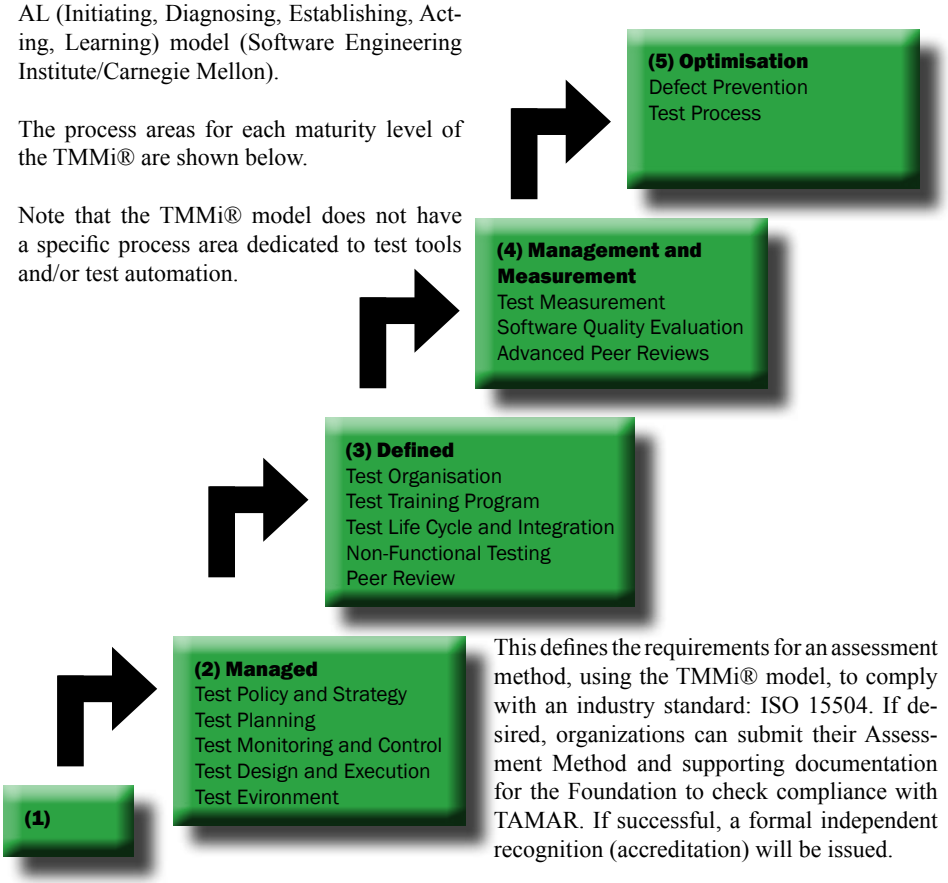
The Emerging TMMi® Model

It was decided to structure the TMMi® model in line with CMMi (staged as the initial deliverable). This was because it would aid understanding and it would seamlessly support and supplement one of the most common Software Engineering Models used globally.

The TMMi® provides a full framework to be used as a reference model during test process improvement. It does not provide an approach for test process improvement such as the IDEAL (Initiating, Diagnosing, Establishing, Acting, Learning) model (Software Engineering Institute/Carnegie Mellon).

The process areas for each maturity level of the TMMi® are shown below.

Note that the TMMi® model does not have a specific process area dedicated to test tools and/or test automation.



Within TMMi® test tools are treated as a supporting resource (practices) and are therefore part of the process area where they provide support, e.g. applying a test design tool is a supporting test practice within the Test Design and Execution at TMMi® level 2 process area, and applying a performance testing tool is a supporting test practice within the Non-Functional Testing at TMMi® level 3 process area.

While it is structured along the lines of CMMi, it draws heavily from many other sources, incorporates proven good principles, best practices and extensive experience from industry practitioners and responds to industry needs.

The preamble and a full definition of TMMi® Level 2 (Managed) are already published. It is anticipated that the full definition for TMMi® Level 3 (Defined) will be available in September 2008 with TMMi® Levels 4 & 5 following by Q1-Q2 2009. While this may not be soon enough for many, the definitions take much effort and are reviewed extensively and rigorously to ensure they are complete and fit for purpose!

Is it enough to publish the Standard TMMi® Reference Model without offering to provide an independent scheme to ensure that organizations have the capability to undertake assessments against the TMMi® model to a robust standard?

We thought not! It is not enough, however, just to publish a model. Too ensure the model is correctly used to assess organizations, the Foundation has also published the TMMi® Assessment Method Accreditation Requirements (TAMAR).

This defines the requirements for an assessment method, using the TMMi® model, to comply with an industry standard: ISO 15504. If desired, organizations can submit their Assessment Method and supporting documentation for the Foundation to check compliance with TAMAR. If successful, a formal independent recognition (accreditation) will be issued.

Following on from this thought process, organizations that have accredited assessment methods can also submit details of Assessors and Lead Assessors to be similarly independently verified by the Foundation as skilled and experienced enough to undertake assessments (and be publicly accredited).

All this provides an independent framework to the industry at large with the means to be publicly recognized as using a robust, repeatable assessment method to evaluate test process capability against a standard reference model (thus allowing easier comparison etc.) using publicly accredited resources – a useable global standard!

The TMMi Foundation

Set up as a "Not for Profit" company registered in Ireland, The TMMi ® Foundation is dedicated to **improving test processes and practice**. The focus of the foundation's activities is the development of a robust model for test process as¬sessment/improvement in IT organizations. We are looking to de¬velop international support and sponsorship from industry leaders.

Its raison d'être is:

1. To develop an International TMMi® Standard for Test Process Maturity Assessments Model, including the develop-

ment of an assessor's scheme and any qualification requirements (including provision of accredited training, examinations and management of accreditation).
2. To facilitate international acceptance of the TMMi® Standard via recognized international bodies and place the standard in the public domain.
3. To promote and support a single, publicly available, international Test Process Maturity Assessment scheme for the benefit of IT testing professionals and Senior Management.

The TMMi Foundation has the following specific objectives:

- Identifying and securing ownership of the TMMi® standard and the ongoing Intellectual Property rights
- A standard TMMi® Model that can be used in isolation or in support of other process improvement models such as the CMMI.
- Assessment Method Accreditation for TMMi® in accordance with ISO15504 and the process to certify commercial assessment methods against the TMMi® Assessment Method Application Requirements (see website).
- An independently managed data repository to support TMMi® accreditation for assessment methods and (lead) assessors and also submitted assessment results.

I would like to emphasize that the work of the Foundation is intended to be available in the public domain, available to all and to provide independent, unbiased services to all.

The current board of directors of the TMMi Foundation come from the UK, Ireland, Holland, Denmark and the USA/India. In addition, there is a rapidly growing list of interested parties across the globe; both individuals and organizations.

How does the Foundation work? There is a Board of unpaid Directors that manage specific areas of our work (technical development, finance, marketing etc.). Funding is through donations and fees for services offered (method and assessor accreditation).

Is it worth it?

It is very well theorizing and postulating to the industry at large that the TMMi® Model is the best thing since sliced bread!

However, the need for a standard Reference Model is being increasingly demonstrated by increasing evidence that there is a fast growing need by the industry at large to be able to evaluate, to a common reference model, test and test-related quality for a variety of reasons including:

- We need to know our weaknesses – improve efficiency and effectiveness of the test function (with ensuing financial and other savings!)
- We need to know where we are (often compared to industry sector benchmarking)
- What do we need to do to gain a certain level of capability/maturity?
- Evaluate capabilities of 3rd party suppliers/managed services (often comparatively as part of tendering process)

This can be greatly enhanced by adopting a standard reference model to measure test processes against which it provides confidence in the assessment ratings (irrespective of assessment providers assuming the method/assessment team is accredited) and allows easy comparison at all levels. These observations are supported by the experiences of assessment practitioners.

It only confirms us in our view that the TMMi® model has the potential to provide the standard by which Test will be judged by - we think it is worth it!

Interested?

The Foundation needs the help of the international community, and we are keen to develop a dialog at all levels.

If you are interested in the products the Foundation is developing, providing constructive feedback and/or in supporting the work of the Foundation, please contact us via our website at www.tmmifoundation.org.

Biography

Brian is a senior IT consultant specializing in Test & Validation and is a recognized thought leader within the industry.

One of his initiatives was to recognize that the test industry needs an industry-wide, fit for purpose, standard Reference Model for Test process capability. He was instrumental in formulating the TMMi Foundation and, as Chair, provides the impetus to its members in their activities as well as strategic direction and management.

He is fully experienced in the definition and implementation of strategic testing solutions for clients. This includes defining, communicating and implementing corporate testing policy & strategy, processes & procedures, standards, test (design) techniques, metrics and monitoring programs, measurement of Return on Investment (ROI), implementing test tools, training, resources and organization and much more. His operational experience covers industry sectors including Banking, Finance, Insurance, Retail and Communications. He also actively understands emerging trends and requirements in the marketplace.

Within his current role at Experimentus Ltd, Brian provides comprehensive, flexible solutions for all aspects of Test & Validation to all levels of a corporation which is both targeted based on priority/risk and achievable while remaining flexible in its approach. This is based on a clear ability to understand the business requirements/drivers, risks and priorities.

Brian has successfully completed the ISEB Foundation & Practitioners Certificates in Software Testing and is a certified trainer of the Foundation course.

Size Does Matter in Process Improvement

by Tanja E.J. Vos, Jorge Sánchez Sánchez and Maximiliano Mannise

Abstract

Based on experiences in implementing testing process improvements in different SMEs, we found that the most common methodologies are aimed at much larger organizations and are therefore difficult to implement in a company profile that has limited resources and low maturity in the testing processes. We therefore propose a series of simple and concrete actions that SMEs can do to spend little resources and get fast results while getting prepared for further, more formal improvement processes.

Keywords: software, testing, SME, process, improvement

1. Introduction

This article describes our experiences we gained during the technology transfer and consulting services projects run with the aim to help enterprises solve a "testing issue" detected by themselves, or just to help them improve their practices in software testing.
In the next few paragraphs we will describe the background and characteristics of those projects.

1.1. ITI (Instituto Tecnológico de Informática)

ITI is a non-profit research institute located inside the Universidad Politécnica de Valencia (UPV) campus. Its main goal is to apply the knowledge obtained from scientific investigation and innovation in state-of-the-art technology in small and medium enterprises (SME) related to information technology and communication (IT).
The institute was constituted by seven research groups, mostly lead by university professors. The experiences described in this article come from the SQUaC group, which focuses on the quality, usability and software certification domain.
ITI's mission includes as a main goal to build bridges between the knowledge and technology acquired through research and the need of simple, practical and robust solutions for the IT-related industry.
In order to reach this goal, ITI supports enterprises with tailored formation, technology transfer, standard solutions and consulting projects.

1.2. The enterprises (SME or microSME)

Many of the enterprises we support are small software companies from Valencia, Spain. Those companies generally share the following characteristics:

- They sell a personalized software product for specific end users
- The software has grown from initially just a small, transparent and easy-to-maintain piece of code programmed by up to two persons to a huge opaque monster impossible to maintain by a group of developers.
- Their customers are used to specifying vague requirements for new developments and having their demands implemented. This usually means having multiple versions of the same software.
- There are no written requirement documents, and if there are any, they have little detail, structure or information.
- There are no structured testing processes nor people specifically dedicated to testing. In fact, many times there are no specific activities related to testing.
- There is little information and management with regard to software bugs. If there is, it usually focuses on bugs detected by customers.

1.3. Their projects (and their problems)

Many times enterprises come to us because they have problems they think they can solve through testing. Problems are mostly the same throughout companies:

- Customers complain because they find many bugs or because the functionality provided does not correspond to their needs.
- Time delays and budgets discussed with the customer are not reached. Testing is often sacrificed because of delays in other parts of the project.
- Bugs, regardless if they are found through testing or reported by customers, are not managed properly, .
- Too many hours are spent on correcting defects and/or misunderstandings in requirements.
- The multiple versions of a product are difficult to manage and/or test.
- There is little or no information about the quality of the products developed.

1.4. Software testing and its levels

For any company (also for SMEs) engaged in software development, whatever the methodology used, testing should be a fundamental part both to ensure that products are developed properly (verification) as that products meet the user's requirements (validation).
In any case, proper testing during the development cycle satisfies ([HMS+07]):

- Testing starts in the early stages of the software life cycle
- You should not only test the final code but also requirements, manuals, documents, etc..
- Testers are involved in requirements retrieval.

The typical levels of testing during an application's life cycle are:

Unit Testing (or components testing). The source software is often divided into units more or less isolated, also called programs, modules, components or classes. Unit testing aims at ensuring that each of these units meets its specification separately. This test is usually done by the same developer who writes the code.

Integration Testing. Once the units have been written and tested the next step is to link them to create a complete system: this is called integration. The purpose of integration testing is to discover defects in the interfaces and interactions between different components or systems.

System Testing. Once the components are integrated and functioning together properly the next step is to consider the full functionality of the system as a whole, which is called system testing. This type of testing is necessary because the lower levels of testing (unit and integration) used a partial view of the system and are not representative of all the actual working conditions. System testing, usually conducted by a team not invlolved in developing the system, can refer to both functional requirements (checking that the system performs the desired functions) and non-functional requirements, such as performance, load, reliability, usability, etc.

Acceptance Testing. The purpose of acceptance testing is to give confidence to the end user that the system will operate as expected. The basis for acceptance testing is the document specification requirements, checking in practice that the system meets all of them properly. This testing is often totally independent of other levels of testing and of the implementation of the system and can be carried out by the end users of the system (with the possible participation of members of the development team).

2. Process improvement and SMEs

2.1. Process improvement methodologies

There are plenty of methodologies and techniques to improve processes in general and testing processes in particular (some of them mentioned in [Swi00] and [Ku06]).

Methodologies such as SPICE [EDM97], developed by ISO / IEC, are intended to be a standard for evaluating software processes in general. Others such as CMMI [CKS03] are even more comprehensive, as they cover all development and maintenance of the product and integrate different areas of knowledge that had previously been treated separately as software engineering, systems engineering or supply.

Other authors have realized that insufficient attention was paid, from these global models to software testing, and developed methodologies for improvement that are focused on that aspect, as Test Improvement Model (TIM) [ESU97] Test Organization Maturity [TOM], Testing Maturity Model (TMM) [BSC96], or Test Process Improvement (TPI) [KP99].

2.2. Process improvement in SMEs, in practice

The main problem with earlier models is that they were originally designed for larger organizations and are not easily adaptable to structures much smaller than our SMEs. The first reason is that the cost and duration of an evaluation process are not compatible with available resources in an SME. In addition, their level of maturity, especially with regard to software testing is usually very low; for example, in a quick initial assessment most SMEs we worked with only reached the minimum TPI level of maturity (level A) in the area of office environment.

Several studies (such as [BJ94]) show that a large number of process improvement projects in SMEs based in models such as CMMI found serious problems, and that a significant percentage of these problems (53%) is related to the size of the enterprise; Success of a project to improve processes increases with the number of persons working on software processes inside the company. As indicated in [LS99] in this type of projects and size of companies, it is impossible to collect enough data to make a solid statistical analysis, being in such cases much more important the human resources, involving key people (those who possess the knowledge) from the beginning.

On the other hand ([Mc00]), the stiffness and formality of these methodologies involve for SMEs an increase in bureaucracy that becomes a liability to the flexibility and innovation that these companies need in a dynamic environment on which they virtually have no control. To decrease the strictness of the methodologies, it is not enough to reduce these disadvantages.

A key factor in the success or failure of such projects lies in the motivation that drives the SMEs to undertake this process of improvement ([BWL98]). For some of them it is "something they must do" faced by pressure from major customers such as government agencies or large companies; in these cases the project is simply regarded one more cost without real improvement expectations. On the other hand, when responsible people of the company think the improvement project is a real opportunity to optimize their internal processes and systems in practice, the chances of obtaining positive results increase.

In the case of Spain studies as [INTECO08] highlight the particular importance of SMEs (representing more than 99% of companies in the software industry) and show that methodologies such as SPICE and CMMI are quite well-known, mainly because of grants from government agencies for its adoption. However, it seems that the companies are more interested in certification for being renamed in front of third parties than in the improvement of internal processes: there is knowledge about the existence of standards and their use outside Spain, but they do not know the details of their implementation in real projects and anything related to its adoption by an enterprise. As a result, the level of adoption is very low. The barriers that SMEs wield are the high costs of adopting the model, the changes in working methods they involve, the need for changes in the organization structure, or the difficulty of evaluating benefits of implementing them.

In short, the experiences of SMEs with process improvement methodologies are often problematic due to excessive investment in time, tools, etc. carried by applying models that are designed for larger organizations with more resources and greater maturity in these procedures. These difficulties become an abyss that separates the SME from process improvement (Figure 1).

3. Preparing the improvement models for SMEs

Given all those problems which arise when applying the improvement models to SMEs, there have been various attempts to adapt or create new models specifically oriented to the

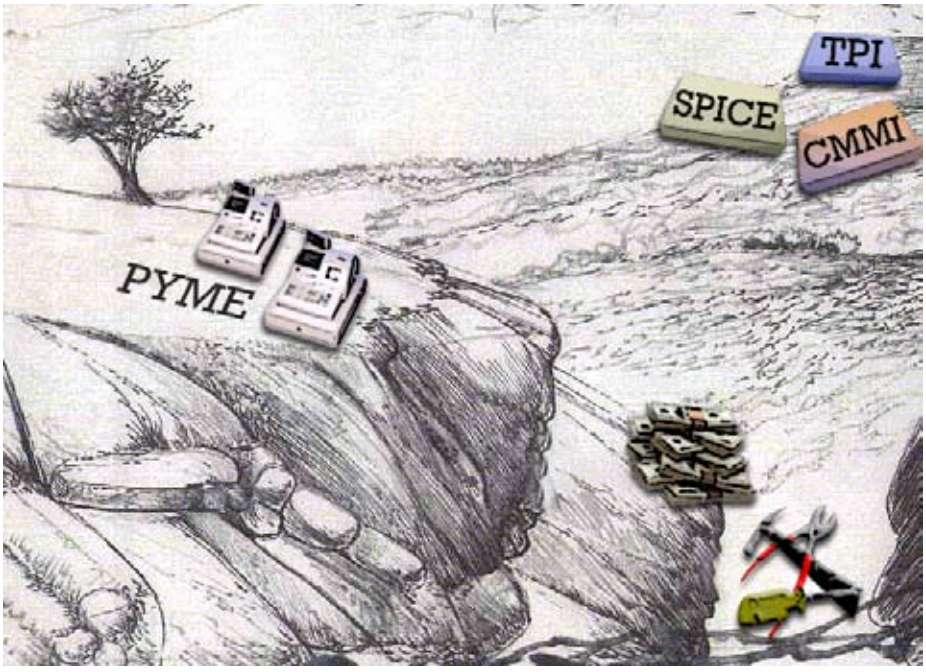


Fig. 1. In the abyss between the SMEs and process improvement methodologies many resources are lost: money spent, tools left over, etc.

Tea testers look at aspects of dried & brewed tea leaves and the liquor

Likewise, PureTesting goes into various aspects & nuances of software testing

We build innovative, end-to-end solutions,
and manage critical testing processes,
and reduce total cost of producing quality software

- Banking & Finacial Services
- Pharmaceuticals
- eLearning
- Datacom & Telecom
- Embedded Systems
- EAI & ISVs

Test Consulting • Testing Services • Testing Products

PureTesting

Testing Thought Leadership

India • USA • UK • NZ

www.puretesting.com

+91 (120) 4621010; info@puretesting.com

Global Software Test Consulting & Services company

characteristics of small businesses.
In general, they consist of simplifications and / or adjustments to more complex models like CMMI (which is used as a reference in [ITMark], especially implanted in Spain) or SPICE / ISO 15504 (which is taken as a basis for *Rapid Assessment for Process Improvement for Software Development*, rap [RTCH00]), or a combination of them (as does [OWPL]).
Other models, such as *Minimal Test Practice Framework* (MTPF) [KRN05] are specifically focused on the activities of software testing and try to adapt to the specific nature of SMEs: scarcity of resources, resistance to internal change, flexibility, and so on.
In Spain [INTECO08], [ITMark] seems to have special significance although there are also initiatives such as the SoftAragón program, which has developed a reduced CMMI model to be applied to Aragonese SMEs, or MESOPYME [CCS+02], specially oriented towards the implementation phase of the improvements.
All these adaptations make improvement models (and, in particular, the testing improvement) more easily applicable to SMEs, but these specific methodologies may not be sufficient, especially because of the lack of maturity and awareness that we found in practice in enterprises. Is there nothing we can do inside the SMEs to ease the implementation of a testing processes improvement methodology? We think there is something we can do.

4. Preparing the SMEs for testing process improvement

Before you start implementing any of the improvement models seen earlier (indistinctly the general models or those tailored to small organizations) in an SME with the characteristics that we saw at the beginning (no formal testing processes, no dedicated staff, etc.), we propose preparing the company with two very practical and simple actions:
1. - Unit testing done by programmers
2. - Staff exclusively dedicated to testing

4.1. Unit testing done by programmers

The sole responsibility of developers in terms of testing would be doing some unit testing on the code they develop, leaving to their choice what to test and how to do it. Let us trust the programmers' knowledge: they know what they want to program, so let them check if they are doing so properly.
The following points are needed in order to succeed with this approach:

- **Force programmers to do unit testing**. After all, the programmers' job is not simply to produce programs but rather to produce error-free quality code. And one way to achieve this is through testing.
- **Motivate programmers to perform testing**. Different techniques can be used. For example, hold regular meetings specifically dedicated to testing ("every Friday morning"); perform informal seminars to explain the progress made ("look what we got/found this week"), or use other means to share that information, as Intranets [LS99] or even Google's approach: testing on the toilet [TotT].
- **Give them freedom to carry out testing as they feel convenient**. In fact, programmers are continuously confronted with new technologies and techniques that they have to use to solve many problems; they constantly seek, learn and apply these new technologies. So why wouldn't they do the same with unit testing frameworks and other tools available?
- **Provide them with necessary resources**. First of all they need time allocated specifically to testing, both to learn and to investigate the best methods and to really do unit testing. Usually they will already have the environment (hardware, facilities, etc.) required for testing because of their normal work. But if the extra effort to realize unit testing is not planned, you will most likely not apply the necessary time.

To complement their knowledge, you may need some kind of training designed to explain in general terms what unit testing is as well as an overview of existing techniques and tools, without going too much into detail, and while leaving the programmers the choice on how to implement this knowledge in their work environment.

4.2. Staff exclusively dedicated to testing

This would involve hiring at least 1 person that is exclusively devoted to high-level testing. His/Her only goal should be finding errors. On top of finding these erros he or she he will also uncover needs in order to carry out his/her work properly (such as better definition of requirements, management defects, risk analysis, planning, etc.) and to help improve the company's development processes.
Ideally, the person responsible for that task would be someone with knowledge and previous experience in software testing. However, taking into account the current job market and the difficulty in finding and recruiting professionals with those requirements (especially SMEs) we can accept a person with no previous experience in testing. In general, it would be someone with a technical profile (e.g. a computer engineer) and certain skills in other areas: communication, planning tasks, etc. In this situation being a self-learner will be very important. In any case, remember that these suggestions seek to lead not to a final testing structure within the organization but rather to a basic preparation prior to undertake a more formal improvement process.
The person assigned to that task could be someone who already belongs to the company, although in practice it is often very difficult for a person who already has prior responsibilities assigned to suddenly stop all his/her previous tasks and focus solely on testing. Hiring a new person avoids this problem but also offsets other problems such as increased costs or lack of knowledge that someone from outside the organization may have.

4.3. Other aspects

In addition to these two points, some basic courses on testing can be useful to increase awareness of the need of it in their business, but too much could be counter-productive if the SME is overwhelmed by the amount of things that they are NOT doing.
We must be prepared because the SMEs are sometimes reluctant to accept that freedom ("let them do what they think they should") at first. However, it is easy to explain that if we start from a situation in which virtually nothing is being done, every step is already a big step on the path to process improvement. It is important that SMEs understand that at this early stage, the most important factor is motivation.
So, after some time (we propose around 6 months), the company will have acquired some knowledge and experience in testing, and we can start with more ambitious plans: to implement a more formal methodology as TPI, more specific courses depending on the needs of SMEs, and so on.

5. Conclusions

Following both advices we present we believe, based on our experience, that an SME can shortly be in a much better situation to successfully undertake a more ambitious testing process improving, such as implementing a more formal improvement methodology, increase testing resources and so on.
In any case, they suppose a relatively small investment and any progress in the maturity of the company on testing-related aspects that is achieved by implementing these practices represents a major breakthrough compared to our usual starting position.
Some of the SMEs we worked with are beginning to implement (partially) these simple tips, and the first feedbacks we get are giving quite positive results. After enough time has passed we will study and present the experiences and results achieved in practice.

References

[HMS+07] HAMBLING, B.; MORGAN, P.; SAMAROO, A.; THOMPSON, G.; WILLIAMS, P. Software Testing. An ISEB Foundation. The British Computer Society, 2007.
[KP99] KOOMEN, T.; POL, M. Test Process Improvement: A practical step-by-step guide to structured testing, Addison-Wesley, 1999.
[Swi00] SWINKELS, R. A Comparison of TMM and Other Test Process Improvement Models. Frits Philips Institute. Technical Report 12-4-1-FP, 2000.
[Ku06] KULKARNI, S. Test Process Maturity Models – Yesterday, Today and Tomorrow. Proceedings of the 6th Annual International Software Testing Conference, Delhi, India, march 2006.
[ESU97] ERICSON, T.; SUBOTIC, A.; URSING, S. TIM-a test improvement model. Software Testing, Verification and Reliability 7(4): 229-246, 1997
[TOM] Gerrald Consulting web site. TOM™ - Testing Organisation Maturity Model http://www.gerrardconsulting.com/default.asp?page=/tomoverview.html

[BSC96] BURNSTEIN, I; SUWANASSART, T.; CARLSON C.R. The Development of a Testing Maturity Model. Proceedings of the Ninth International Quality Week Conference, San Francisco, 21-24 may, 1996.
[CKS03] CHRISSIS, M. B.; KONRAD, M., SHRUM, S. CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley, 2003.
[EDM97] EL EMAM, K., DROUIN, J.N.; MELO, W. SPICE: The Theory and Practice of Software Process Improvement and Capability Determination. Wiley-IEEE Computer Society Press, 1997.
[INTECO08] Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España. Instituto Nacional de Tecnologías de la Comunicación, april 2008.
[BJ94] BRODMAN, J. G.; JOHNSON, D. L.., What Small Businesses and Small Organisations say about CMM? Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italia, may 1994.
[LS99] LIED, H.J.; STLHANE, T. Experience from process improvement in a SME, EuroSPI'99, 1999.
[Mc00] McADAM, R. Quality models in an SME context: A critical perspective using a grounded approach. International Journal of Quality & Reliability Management, Vol.17, Issue 3, pp. 305-323, 2000.
[BWL98] BROWN, A.; WIELE, T.V.D.; LOUGHTON, K. Smaller enterprises' experiences with ISO 9000. International Journal of Quality & Reliability Management, Vol. 15, Num. 3, pp. 273-285(13), 1998.
[OWPL] OWPL Software Process Improvement for VSE, SME and low maturity enterprises; version 1.2.2. FUNDP-CETIC – Software Quality Lab – 2000-2006. http://www.cetic.be/article393.html
[ITMark] ESI web site. IT Mark. http://www.esi.es/index.php?op=15.1.2
[KRN05] KARLSTRÖM, D.; RUNESON, P.; NORDÉN, S. A Minimal Test Practice Framework for Emerging Software Organisations. Software Testing, Verification and Reliability, VL 15, NO 3, PG 145-166, John Wiley & Sons, Ltd, 2005.
[RTCH00] ROUT, T.; TUFFLEY, A; CAHILL, B; HODGEN, B. The Rapid Assessment of Software Process Capability. Proceedings of SPICE 2000, 2000.
[CCS+02] CALVO-MANZANO, J. A.; CUEVAS, G.; SAN FELIU, T.; DE AMESCUA, A.; GARCÍA, L.; PÉREZ, M. Experiences in the Application of Software Process Improvement in SMES. Software Quality Control 10, 3 (Nov. 2002), 261-273.
[TotT] Google Testing Blog. Introducing "Testing on the Toilet". http://googletesting.blogspot.com/2007/01/introducing-testing-on-toilet.html



Biography

Tanja Vos is director of the SQUaC (Software Quality, Usability and Certification) research group of the ITI (Instituto Tecnológico de Informática). She has a master in Computer Science and a PhD on verification of distributed systems, both from the University in Utrecht. Tanja has worked as a researcher and professor at various universities like the University of Utrecht (The Netherlands), Universidad Mayor de San Simón (Bolivia), University of Cambridge (UK) and the Mediterranean University of Science and Technology in Spain. Dr. Vos has participated in research projects funded by the European Commission, the Spanish government or by industry. She has more than 10 years of experience in the area of software quality and testing.

Jordi Sánchez is an engineer in Computers Science by the UIB (University of Balearic Islands); currently he works at the SQUaC (Software Quality, Usability and Certification - http://squac.iti.upv.es) group of the ITI (Instituto Tecnológico de Informática), in Valencia, on projects related to software quality: testing, usability, accessibility, etc. Previously he has worked for more than 10 years in the financial sector, studying and applying new technologies to collaborative environments in Intranet and Internet, always from the point of view of user experience. He also writes on his personal web page, http://jordisan.net

Maximiliano Mannise is member of the SQUaC (Software Quality, Usability and Certification) research group of the ITI (Instituto Tecnológico de Informática). He had a degree of Computer Engineering from the Universidad Católica del Uruguay. He worked nine years at IBM Uruguay, over two years as a Test Project Manager. Since 2007, he is with ITI and works as a consultant in process and software testing improvement and as a researcher in the area of software testing. Maximiliano is interested in practical test process improvement applied to software organization.

Personnel evaluation based on tests

© iStockphoto

by Andrey Konushin

It can be asserted that software testing as an independent technological discipline appeared in the natural evolution of program engineering knowledge. Russia, with a noticeable lag, repeats processes which are taking place in West. America, for example, has already realized the need for "independent testing". Over time, western specialists have begun to treat quality problems with much more responsibility. After years of evolution, testing had been transformed into a rather structured technological process that requires not only human resources, but also special software and hardware which cannot be assumed as available to every company. Apart from that, testing is a specific science that requires specific qualifications. The experience is that in Russia today such qualifications are rarely taken. Even in the West there are not many educational institutions that teach testing. In Russia they are absent. And not every programmer is able to become a good tester [1].

In the Russian provinces the situation with personnel selection is much more difficult than in the larger cities. This is principally the result of low standards, the evolution of the testing branch and a lack of teaching organizations. In spite of this, there are periodical seminars and trainings for software testers, all of which are in part non-standard because the trainers lead the seminars using their own subjective program. Thus when selecting the personnel in these provincial areas department leaders are faced with the question of how to evaluate an employee when he/she has no experience or knowledge in testing. Not only is the current qualification level of the candidate important here, but also his/her professional status. Having experience means you can teach employees based on your own strengths. A personal evaluation which can be assessed in all testing areas can help here. However, such evaluations at this stage only provide a way to obtain information about a candidate. Evaluation of this data is still an expert task with low levels of formalization. Results depend on the experience and professional instincts of the test manager.

Suppose that a recently hired employee has no knowledge in testing or the knowledge is not systematic and based purely on previous experience. This can result in problems in teaching this employee.

The International Software Testing Qualifications Board (ISTQB) is now just starting to pick up speed in Russia and is set to become one the main trends in the training and certification of software testers. Training programs are independent from vendors. More than 120 professionals in software testing from all over the world took part in the development of syllabi, which gives them a high level of reliability. Apart from that, the syllabi are international and generally accepted: they exist in more than 20 countries and several tens of thousands of specialists are certified [2]. The certification scheme provides qualification based on exams.

There is a good reason to evaluate personnel periodically. Moreover, evaluations need to be processed for both the new and existing testers. World trends show that evaluation is assumed to be in the form of exams [3]. A properly constructed exam permits an objective evaluation of qualification levels shown as a quantitative ratio. It is a laborious task to conduct such exams; maybe even more laborious than an expert assessment of qualification levels. But exams have an advantage: they can be applied more than once and to multiple employees concurrently. This can save precious manager's time, as shown by the following calculations.

Assume that expert evaluation and exam evaluation give the same result – objective knowledge of the employee's qualification level. Time spent on expert evaluation equals the time spent on evaluation by exam.

The formulas are based on the assumption that employees are rated on an hourly basis. The time spent by an examiner on performing direct tasks and the time spent on references cost the same.

Expert evaluation cost:

$$EC_E = \sum_{0}^{N_T} (R_E + R_T)$$

where - cost of expert hour of personnel evaluation work, - cost of examiner hour of work, - number of testers being evaluated.

Test evaluation cost:

$$EC_T = R_E + \sum_{0}^{N_T} R_T$$

where - cost of exam development by expert, - cost of examiner hour of work, - number of testers being evaluated.

Suppose that a testing department consists of 10 testers. Evaluation is held twice a year. The cost of examiner work is 14 €. The cost of an expert is 22 € per hour. Evaluation of one employee takes 1 hour in both variants considered. The expert is able to develop an exam in 5 hours.

$$EC_E = \sum_{0}^{10}(22+14) = 10*36 = 360EUR$$

$$EC_T = 5*22 + \sum_{0}^{10}14 = 110+140 = 250EUR$$

Of course, results processing should be automated as much as possible. The time spent on automation is not taken into account in the formulas shown above. It is possible to assume that automation consists of the elementary report on the number of correct and incorrect answers. If we assume that one exam can be used twice (e.g. twice a year or two exams alternating over two years) and evaluation is needed twice a year, then we have the following results per year.

$$EC_E = 2*\sum_{0}^{10}(22+14) = 2*10*36 = 720EUR$$

$$EC_T = 5*22 + 2*\sum_{0}^{10}14 = 110+280 = 390EUR$$

Thus, with some assumptions, calculations show that the most effective method of evaluation is exams, other conditions being equal. In this article a software testing department is considered. However, it is easy to see that these calculations are applicable to different key industries that face the problem of evaluation of personnel.

1. "CIO" magazine №10, 19.12.2002, http://www.cio-world.ru/offline/2002/9/22721/
2. International Software Quality Specialists Certification Scheme, A. Konushin, V. Vershinin, Contemporary problems of economics and new research technologies: interuniversity collection of proceedings. Part 1 / VZFI Vladimir Branch. – Vladimir, 2006. ISBN 5-93350-126-3
3. Personnel evaluation methods, http://www.management.com.ua/hrm/hrm029.html

Biography

25 years old. Specialist in software testing. ISTQB Certified Tester, Advanced Level.

Master of techniques and technologies in informatics and computer engineering, Vladimir State University, Russia.

Since 2004: Test Manager of Inreco LAN, a software development company (http://inrecolan.com). His main research interests include information management and system analysis both in software testing and related fields.

Since 2006: Lecturer of Information Systems & Information Management chair of Vladimir State University. In 2008 Andrey developed and implemented the "Software testing" course for the University based on ISTQB Syllabus.

Andrey is one of the founders of the Russian Software Testing Qualifications Board. Since 2006 he is the President of RSTQB.

Yaron's Forecourt

Practicing Test Techniques - A Discussion

by Yaron Tsubery

Dear colleagues and readers,

Welcome back to our joint forecourt, I hope that you're already equipped with a glass of chilled wine or a pint of cold beer… or what ever you choose in order to relax, willing to read, and to share your thoughts on today's subject…

Test Techniques. Does just hearing the expression make you feel uncomfortable? Or are you still relaxed? Do you feel secure discussing Test Techniques? Or do you dismiss it as "not for us"? When I find myself speaking about Test Techniques with my employees (especially newbies – no offence…) or even sometimes with my managers (who are R&D managers or Business managers), they look at me as if I just fell from the sky… It's clear which camp they are in – they are uncomfortable talking about them.
Why do you think this is so? What is it about Test Techniques that makes it so "terrifying"? Well, I haven't done a thorough research on this subject from the psychological angle, but let us look at some practical aspects related to Test Techniques that I've been engaged with.

Yes, you, who don't feel comfortable with Test Techniques or are "terrified" by them, you should be. It is not so easy to understand them, it is not easy to understand how to implement them in a real project and it is not easy to estimate how much time it will take. It takes time for practitioners to get use to the fact that for each project they must spend time, effort, and stimulate discussions on Test Techniques, while you – the "terrified" people – probably don't even consider them in your plans!

Well, if you just think that it's enough to go to training, learn the theory from the books, and get to know some practical aspects from your professional and skilled tutors who will give you only a clue about it, you're partly right. They will give you the theory, but you'll not be in the real world.
The real world is where you have to make this effort part of your plan, and then justify this plan to a bunch of unconvinced peers and managers. The first time you do that, you have no proof that this effort will buy you anything. You may need to get used to see lots of gloomy faces of your peers and manager/s who are convinced you are just spending their time and money.
Yes, the time will come and you'll be able to prove the message of the cost effectiveness of investing time on Test Techniques, you'll be able to show the gaps between your test coverage before and after, which tests caught the most important bugs in the shortest time. Yes, it will be hard in the beginning, there will be doubts all around – even you will have some (I had once, in a very stressed project) – don't give up, keep on with your mission!

So, what does it take to use test techniques? Firstly, you have to have the knowledge and skills to implement these techniques. This is no small feat. It's not enough to learn the techniques; you have to have implemented them and understood their benefits. Secondly, you need the understanding that we must invest time in the analysis phase, to come up with the best test coverage for the project. Thirdly, you need to be wise in how to communicate it with your peers!

Well, at the end of the day, dealing with Test Techniques is also a matter of maturity – in order to reach the high quality that we require from the products our company produces, we need to have a mature testing organization too! Speaking the same language – "engineering" – with our peers, and showing test coverage by using the best test techniques for each area/domain, will enhance the maturity of the product, and achieve respect for the test organization, and recognition of its maturity.

Summarizing this short column, I would like to open it for discussion by asking few questions which came up while writing this paper and before…: Do we use test techniques in our organizations? Do we have the knowledge? Can we estimate and later measure the profit of using test techniques? What is the impact on our daily job and on the test organization maturity? Can you share and advise how to leverage the use of test techniques? Are there experts who can support using test techniques, or even oppose (with proven reasons)?

Well, that's it for today. I've had my couple glasses of red dry wine; I hope that you enjoyed the time as well. I'm sure now that even if you're relaxed, you still have enough strength to share your knowledge and experience; please don't think twice, your inputs are important for others in the forecourt, and I'll be more thankful if you'll send them to the following account, prepared especially for our discussions: yaron@testingexperience.com

Yours truly,
Yaron

Biography

Yaron Tsubery has been a Director of QA & Testing Manager at Comverse for seven years. Yaron is in charge of a large group that includes Testing Team Leaders and Test Engineers. The group deals with Functional, Non-functional, Load & Performance tests, some of which are done off-shore. The group has much experience in managing and controlling acceptance tests. Yaron is also known as a manager who has built testing groups in his past and also acted as a consultant to testing managers while they were building their testing groups.
Yaron has been working in software since 1990, and has more then 15 years in testing field as a Test Engineer, Testing TL, and Testing Manager. His original profession was Systems Analyst.
Yaron worked in Product Management, Project Management and Development for at least 3 years before becoming a Director of QA & Testing.
Yaron is a member of ISTQB (International Testing Qualifications Board – www.istqb.org) and is the President and founder of the ITCB (Israeli Testing Certification Board – www.itcb.org.il). He is a member of IQAMF (Israeli QA Managers Forum) and in SIGiST Israel.
Yaron has been invited as a speaker to some important international conferences to lecture on subjects related to testing.
Yaron has lots of experience in banking business processes. For the last 7 years he has implemented best practices in a field where he is in charge of producing complex systems for telecommunication companies such as Vodafone, T-Mobile, NTT DoCoMo, Verizon Wireless etc';
Main systems in Mobile technology: i-mode, Mobile GW and Billing Proxy, in SI projects: Content Data Management solutions, Video Solutions, IPTV and IMS.

What about your intellect?

CaseMaker – the tool for test case design and test data generation

If you are ISTQB Certified Tester, you can get a **free version** of the **new** client software for **12 months**.
Please register at www.casemaker.eu/istqbct

ISTQB
Certified Tester?
Get a FREE
Version

CaseMaker®

www.casemaker.eu