13

March 2011

# testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

# Testing @ Domains –

## How does Finance, Automotive, Medical etc test? Do we have to take care of the domains?

Conferences Special

iqnite DEUTSCHLAND 2011

# Certified Agile Tester

Pragmatic, Soft Skills Focused, Industry Supported

CAT is no ordinary certification, but a professional journey into the world of Agile. As with any voyage you have to take the first step. You may have some experience with Agile from your current or previous employment or you may be venturing out into the unknown. Either way CAT has been specifically designed to partner and guide you through all aspects of your tour.

The focus of the course is to look at how you the tester can make a valuable contribution to these activities even if they are not currently your core abilities. This course assumes that you already know how to be a tester, understand the fundamental testing techniques and testing practices, leading you to transition into an Agile team.

The certification does not simply promote absorption of the theory through academic mediums but encourages you to experiment, in the safe environment of the classroom, through the extensive discussion forums and daily practicals. Over 50% of the initial course is based around practical application of the techniques and methods that you learn, focused on building the skills you already have as a tester. This then prepares you, on returning to your employer, to be Agile.

The transition into a Professional Agile Tester team member culminates with on the job assessments, demonstrated abilities in Agile expertise through such forums as presentations at conferences or Special Interest groups and interviews.

Did this CATch your eye? If so, please contact us for more details!

## Book your training by Díaz & Hilterscheid!

Open seminars in Berlin:
May 16 - 20, 2011
June 6 - 10, 2011
August 22 - 26, 2011
October 10 - 14, 2011
December 5 - 9, 2011

Díaz & Hilterscheid GmbH / Kurfürstendamm 179 / 10707 Berlin / Germany
Tel: +49 30 747628-0 / Fax: +49 30 747628-99
www.diazhilterscheid.de training@diazhilterscheid.de

Díaz Hilterscheid

# Editorial

Dear readers,

The first issue of 2011 drives us crazy! We have received a great deal of articles and decided to issue many of them. Exactly 32! This seems more a book than a magazine...

What happened is that a well known company sent the call for papers to the internal testing group, which we really appreciate a lot. This caused that many of their testers handed in a paper. If we wanted we could issue more than one magazine with only these papers. Please understand that we can not do this, even if most of the papers are very interesting. This is a sign, though, of the good quality of the testers and also about the communication and collaboration within the company culture. I like it. The huge amount of work is done mostly by our lector and graphic designers.

This year started quite strong for us, especially for me, being the Senior Editor of our Testing Experience magazine, but as well in the other parts of the company such as consultancy, training and our events.

New is that we are one of the first training providers for CAT – Certified Agile Tester. We just informed a few customers to run pilot trainings – 50% of the training is practical – we got 6 trainings within a week to get done: Germany, Belgium, USA! It seems that CAT is going to run well. If your company is interested in being one of the first training providers, please contact iSQI at www.agile-tester.org. The world is big enough for all!

Also, if you are interested in being trained, then send us an email!

The Belgium Testing Days were amazing. I don't know if you followed it via twitter (#btd11). It was a very good conference and we are very happy that we organized it together with AQIS – our local partner in Belgium. We are looking forward to run it next year. Don't miss it!

Gojko Adzic asked us to run a new conference. It is called AgileReq and it is the conference for Product and Requirements Management in Agile. It is going to be in London on September 9-11, 2011. One of the innovations regarding this conference is the time period: the tutorials run on Friday and the conference on the weekend. We think that this makes the conference attractive to the attendees and the companies of the attendees! Gojko is the Program Chair. We are already counting on support by Ellen Gottesdiener, Christian Hassa, Gojko Adzic, Dan North and Chris Matts. Follow it via Twitter (#agilereq) The call for papers has already started!

Alex Collino did a great job as Program Chair of the Agile Testing Days 2011. We have a very interesting, international and interactive program. Have a look at the magazine ad and at the website (www.agiletestingdays.com). We are lining up many new keynote speakers. It looks like the Agile Testing Days is becoming a female driven conference. We have again 6 female keynote and tutorial speakers ;-) Anyway the line up of speakers is amazing: Johanna Rothman, Esther Derby, Lisa Crispin, Janet Gregory, Linda Rising, Lit Keogh, Michael Bolton, Gojko Adzic, Jurgen Appelo and Lasse Koskela. Don't miss it and follow it via Twitter (#agiletd).

The Testing & Finance 2011 (www.testingfinance.com) will take place in Bad Homburg on 9 and 10 of May, 2011 near Frankfurt. This is the conference for the testing professionals in the finance industry. Don't miss it!

We are also sponsoring the Ignite conference in Düsseldorf. This is one of the biggest testing conferences in Europe!!

Last but not least thanks again to all speakers, sponsors and readers helping us to improve and to issue a great magazine.

I would really appreciate if you could forward the link of the magazine to your interested contacts and friends.

Enjoy reading!

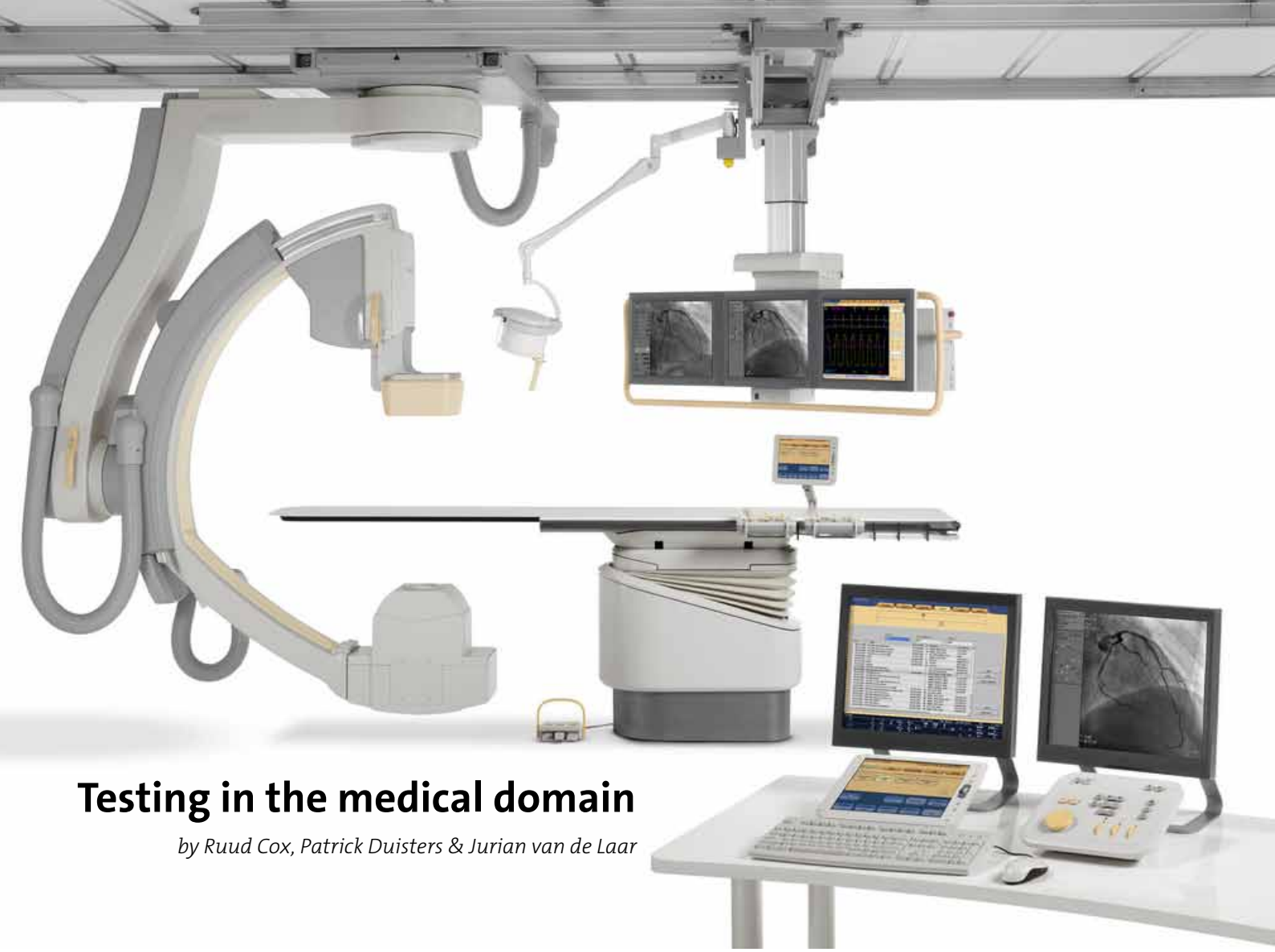
José

# Contents







# Testing in the medical domain

*by Ruud Cox, Patrick Duisters & Jurian van de Laar*

*What is the influence of the medical domain on the way we organize our testing? Are there any typical constraints on our test approach imposed by the domain? What are the current trends and new insights regarding testing in the medical domain? Are there lessons we learned in this domain that would also be useful in other domains? These are the questions we will address in this article. The authors of the article all have experience in various testing roles in the medical domain.*

For testing in the medical domain it may seem obvious to choose a formal approach because of the safety critical nature of the applications. The risks related to systems in this domain typically go far beyond financial risks, as risks can be hazardous or can cause permanent harm to people. That's one of the reasons why regulatory bodies keep track of product quality and of the development process, production and service.

We have several years of experience in the area of integration and testing of large and complex imaging systems used in hospitals all over the world to make X-ray images of the heart and vascular system of a patient for and during medical examination. In these systems complex algorithms are implemented in software (both embedded software and system software) to control the examination process, acquiring the X-ray images but also to control the movement of the X-ray tube/detector combination and the table the patient is lying on while taking into account all surrounding equipment. Everything should work together while the patient's safety as well as the safety of the doctor and other personnel may never be endangered.

The functionality of the system must be verified as well as validated. In this context 'verification' means providing objective evidence (e.g. to regulatory bodies) that specified requirements for this product actually have been fulfilled. Objective means factual, independent of someone's judgment or opinion. In other words proving - with factual test results - that the system works as specified. 'Validation' means that - apart from specifications - the system works in the hospital environment in the way that users intend to use the system. Therefore validation also implies clinical evaluation, which means that the efficacy of the system must be actually proven in a hospital environment. Besides *functionality* there are also other quality characteristics that are important when testing systems in this domain. In our experience usability, reliability, safety and security are the most important additional quality attributes for these systems.

It is typical for the medical domain that objective evidence has to be gathered by testing. This evidence has to be provided to regulatory bodies like the FDA (Food and Drug Administration of the American government). The influence of bodies like the FDA is very large, because an organization like the FDA has investigative jurisdiction and the authority to inspect and stop the export of medical devices to the United States if these devices do not comply with their regulations.

For providing this objective evidence, the tester can resort to various instruments:

1. **Risk based testing**: Like in any large and complex system, testing 'everything' is infeasible, even for safety critical systems. Actually, and *especially* for safety critical systems, it is important to apply a thorough risk analysis to ensure that the most risky parts of the system are tested the most thoroughly and with the highest priority. In our experience the PRISMA® methodology (Product RISk MAnagement) is often applied. With this approach the risk items (that can be features or requirements) are assessed in terms of likelihood (the chance that a defect might occur) and impact (the consequence of a defect if it occurs). Based on these scores the risk items are plotted in a two-dimensional 'risk matrix'. Formal records of this risk analysis are kept and the results are laid down in the (master) test plan. Based on this risk matrix a substantiated test approach is developed, differentiating in e.g. thoroughness of applied testing and review techniques.

2. **Test approach**: Based on the results of a risk analysis, the tester can apply a differentiated approach to derive tests from the test basis. Depending on the risk level he can choose from various formal or informal test design techniques. The level of formality of a test technique is determined by the extent to which the process of deriving test cases from the test basis is prescribed. For very formal test design techniques, like e.g. Decision Table Testing, all the steps from finding the test conditions in the test basis to the specification of test cases and test procedures are exactly specified. Applying such a technique is like following a recipe. In this respect, the design process becomes less dependent on the person (the tester), and the resulting test cases guarantee a certain type and level of coverage. Decision table testing, for example, guarantees that all possible combinations of the input conditions have been exercised. The process of deriving tests will be more objective. Finding the inputs (test conditions) will of course remain very strongly dependent on the skills of the tester.

3. **Traceability**: Regulatory bodies like the FDA, but also maturity models like CMMI (Capability Maturity Model Integration) require a certain level of traceability. The pitfall for many organizations - not only in the medical domain - is that they try to achieve traceability to an almost exhaustive extent, which increases the amount of work - and maintenance - exponentially. From a testing point of view the most important objective of traceability is to demonstrate (provide evidence) that requirements - typically at higher levels, e.g. system level - are exercised by related (system-) test cases.

The influence of regulatory bodies on an organization in the medical domain is quite large, especially on the testing process, for the purpose of 'providing evidence' as explained above. What are the consequences of this influence? Does this influence contribute to 'better testing' or not? We think that this influence has both positive and negative effects. The obligation to adhere to regulations can be used as enabler to improve the test process. It provides focus in the organization to deliver quality and it helps to bring the testing activities to a higher maturity level - simply because this is necessary for providing the required objective evidence. On the other hand, the risk of an FDA finding - that may lead to serious consequences for the business - is constantly lurking. Because the consequences may be so severe, it is vital for the organization to take all necessary measures to prevent this risk becoming reality. This may result in a very sensitive, risk-averse attitude. Just to be on the safe side, processes and measures may be defined or interpreted more strictly and formally than what the regulatory bodies actually require. In the following, the process may become a goal in itself, limiting the creativity and efficiency of the development organization.

In our practice we sometimes see misunderstandings and pitfalls due to this attitude. These are some examples:

- "**Testing everything after all**": Even after a risk analysis has been done, some testers still prefer to execute all available test cases instead of applying a differentiated test approach. 'Executed all available tests' is interpreted as 'Tested everything'. Especially large test suites provide a false feeling of reassurance. A large amount of tests doesn't necessarily mean high coverage and even the highest requirements coverage or code coverage do not guarantee that all defects will be detected. Testing everything is infeasible.
- "**Formal test techniques will result in higher quality**": Untrue. Formal doesn't necessarily mean that the test is better. Informal tests are often even 'smarter' because the tester uses his skills and experience to detect the most important defects.
- "**Regulatory bodies like FDA prescribe how testing must be done**": No. FDA does neither prescribe the test approach nor does it prescribe that formal test techniques should be applied. The FDA does require transparency: demonstrate (document) how the requirements have been covered and which test results you have used to determine whether the test passed or failed. It's important to show (document) what you have done and why (rationale), but the approach itself can be formal as well as informal.
- "**Thorough testing means repeating the same tests over and over again**": This is another misunderstanding. Repeating the same tests may result in 'the pesticide paradox'. When you always use the same pesticide (test suite), you won't find any new defects anymore, because the 'bugs' have become resistant for the cure.

As a next phase in the evolution of the test profession becoming more mature - probably as the result of the need for higher efficiency and more effectiveness - we see a trend to more agile and informal practices being applied in the medical domain.

Applying an approach like Exploratory Testing (ET) can be very beneficial - also in the medical domain. As mentioned above, provision of objective evidence is needed. While applying ET the common properties of ET provide support. If charters are made (and reviewed) in advance and sufficient logging is created during execution, ET can successfully be used in the medical domain.

Whereas formal techniques attempt to achieve a certain level of coverage by the way test cases are derived from the test basis (the 'recipe'), informal techniques require another kind of 'evidence'. In this case, the competence of the tester makes the difference!

Also on this issue, the regulatory bodies require evidence: resumes ('curriculum vitae'), records of test related training, and, where applicable, certifications of the testers are to be kept and can be used to convince the regulatory bodies of the skills and experience of the tester(s) involved.

What about traceability? The contents of the charters can be related to the tagged requirements. During execution the require-

ments can be referred to in the notes and system logging or video can be tagged with the requirements.

Of course, this requires some effort, but compared to formal techniques it is a rather limited effort.

Even the risk of a 'pesticide paradox' is reduced significantly by Exploratory Testing. The time saved compared to formal testing can be spent on finding new or 'other defects', defects that might not be found by formal test design techniques which are based on specified requirements.

Our conclusion is that testing in the medical domain can benefit from both formal and more informal and agile practices. We have published this article because we are convinced that other domains can benefit from these insights as well. The importance of testing in the medical domain is driven by the need for providing evidence to regulatory bodies as well as the crucial importance of developing safe products. These have been the driving force to become leading in test process maturity. The lessons learned, as described in this article, are not only useful in this domain.

Furthermore we see that the need for risk management and 'control' is also growing in other industries. In domains where software controlled safety critical systems operate this isn't a surprise. However, in our experience there are not always regulatory bodies, like the FDA, which have the authority to inspect the processes in such detail and with such consequences. In the rail industry, for example, there are regulations, but regulatory bodies don't always have authority similar to the FDA - at least not in all countries. Often regulatory organizations rely on the evidence delivered by the manufacturers. The (testing) process is not inspected on site.

We also have seen examples where independent specialized companies are asked to perform expert reviews or audits on the testing of safety critical (software) parts of the technical systems for tunnels. In these cases the governmental safety & security officers rely on the knowledge of the involved experts.

Finally, we expect that the influence of regulatory bodies will increase - also in other domains. We therefore think that the lessons learned in the medical domain may become more and more valuable to others as well. A starting point can be to just write down what you do, and do what you have written down.

## > biography

**Ruud Cox**
*is a Senior Test Consultant at Improve Quality Services. He spent the last three years working in the medical device industry as Test Manager. He is one of the founders of the Dutch Exploratory Workshop on Testing (DEWT).*

**Patrick Duisters**
*has for over 10 years worked within quality and testing in the ICT industry. Patrick has extensive experience in software testing of both administrative and technical systems. He now works as test consultant, test architect and test process auditor at Improve Quality Services. He was responsible for establishing and maintaining quality systems of test organizations and for testing systems for financial services and technical medical systems. Furthermore, Patrick has experience in FDA audits as well as auditing safety critical tunnel systems. Patrick is ISEB Practitioner in Software Testing, TMap Next, and Prince2 certified. He is an accredited teacher of the ISTQB® Foundation and Advanced Level, and also teaches TMap Next Foundation and Advanced level, test design techniques in practice, and reviews & inspections. Patrick is accredited by the TMMi Foundation as (lead) auditor.*

**Jurian van de Laar**
*has practical working experience in software engineering, team leading, software quality and testing since 1994. He graduated in Computer Science and is specialized in testing and quality improvement. Jurian is Senior Consultant at Improve Quality Services and consulted companies like InTraffic, TomTom, Philips, Triodos Bank and DHL. He is an accredited lead assessor TMMi and had a leading role in achieving TMM Level 2 for a large company in the medical domain. Jurian is teacher of courses in reviews and inspections, CMMI and the training programs for certified professionals in requirements engineering (IREB) and software testing (ISTQB). He is certified in Prince2, TMap, ISTQB and IREB. Jurian is a regular speaker at (inter-)national conferences (EuroSTAR 2009, Swiss Requirements Day 2010, Belgium Testing Days 2011).*



# EUROPE 2011

Testing & Finance

May 9–10, 2011
Bad Homburg v. d. H., Germany
The Conference for Testing & Finance Professionals

www.testingfinance.com

| Time | Track 1 | Track 2 | Track 3 | Track 4 | Vendor Track |
|---|---|---|---|---|---|
| 08:00 | Registration | | | | |
| 09:10 | Opening Speech<br>José Díaz | | | | |
| 09:15 | **Keynote:**<br>**"Social Banking 2.0"**<br>Lothar Lochmaier (Textbüro Lochmaier) | | | | |
| 10:15 | Break | | | | |
| 10:20 | **"Quality Approach at 'Zurich Center of Excellence' in Barcelona"**<br>Javier Fernández-Pello & Raul Alares (Zurich CoE) | **„Effizientes Testen"**<br>Robin Schönwald (SAP Deutschland) | **„Ökonomische Risikogemeinschaft – Probleme und Lösungsansätze bei der Umsetzung innerhalb eines Konzerns"** | **„Identity Management im Umfeld von IT-Sicherheit und IT-Governance"**<br>Frank Pevestorf (Díaz & Hilterscheid) | |
| 11:10 | Coffee Break | | | | |
| 11:30 | **"Test Process Evolution – How Far Do We Go?"**<br>Chris C. Schotanus (Logica) | **„Modellbasiertes Testen (MBT) zur Erstellung wartungsarmer, automatisierter Tests in Ergänzung zu statischen Kode-Analysen"**<br>Michael Averstegge & Jan Mikus (CS Consulting) | **„Methoden der Ertragsmessung in Banken"** | **„Viele Wege führen zum Identity Management – welcher ist der Richtige?"**<br>Jutta Cymanek (Omada) | **„Managed Testing Services – Ein Erfolgsmodell für Groß und Klein"**<br>Frank Schmitz (Steria Mummert Consulting) [1] |
| 12:25 | **Keynote:**<br>*TBD*<br>Achim Wagner (Dekabank) | | | | |
| 13:25 | Lunch | | | | |
| 14:40 | **"Using the TPI-next Model for Test Process Improvement"**<br>Graham Bath (T-Systems) | **"Interfacing Interface Testing"**<br>Patrice Willemot (CTG) | **„Aktuelle Änderungen im Aufsichtsrecht"**<br>Dr. Karl Dürselen | **„Data Loss Prevention: Welche Gefahren bestehen, wie kann das Risiko minimiert werden?"**<br>Benedikt Jäger (FMC Feindt Management Consulting) | |
| 15:35 | Coffee Break | | | | |
| 15:45 | **"Role of Negative Requirements in Security Testing"**<br>Nageswara R. Sastry (IBM) | **„Tagebuch eines nächtlichen Tests"**<br>Achim Lörke (Bredex GmbH) | **„Liquiditätsvorgaben der Aufsicht"**<br>Dr. Thomas Dietz (Fachhochschule der Deutschen Bundesbank) | **„Identity & Access Governance (IAG)"**<br>Dr. Martin Dehn (KOGIT GmbH) | **„Einführung der neuen Eigenkapital vorschriften (CRD II) bei knapp 800 Banken"**<br>Mario Berger & Christian Sattler (CGI) [1] |
| 16:35 | Coffee Break | | | | |
| 16:50 | **"Test Manager Ability to Lead"**<br>Thomas Axen (ATP) | **"Software Testing Banking Applications for Mobile Platforms such as IOS and Android"**<br>Christian Ramírez Arévalo (Certum) | **„Herausforderungen und Implikationen einer modernen Kreditrisiko modellierung – Theorie und gelebte Praxis in einer Zentralbank"**<br>Dr. Jadran Dobrić (WGZ BANK) | | |
| 17:45 | **Keynote:**<br>**"Industrial Espionage (at its best)"**<br>Peter Kleissner | | | | |
| 18:50 | Social Event: Dinner & Theater | | | | |

1

# Conference Day 2
May 10, 2011

EUROPE 2011
Testing & Finance

| Time | Track 1 | Track 2 | Track 3 | Vendor Track |
|---|---|---|---|---|
| 08:00 | Registration | | | |
| 09:10 | **Keynote:**<br>**„Basel III und CRD IV: Die komplexen Neuerungen zum Eigenkapital und deren Auswirkungen auf die Risikotragfähigkeit der Banken“**<br>Prof. Dr. Hermann Schulte-Mattler | | | |
| 10:15 | **“The New Philosophy: Preparationism, When The Past Just Won’t Do”**<br>Jamie Dobson & Jorrit-Jaap de Jong (Ugly duckling software) | **“We Rebuild the Bank”**<br>Matthias Leitner (IngDiba) & Helmut Pichler (ANECON) | **„Schwerpunkte der dritten MaRisk-Novelle”**<br>Dr. Ralf Hannemann (Bundesverband Öffentlicher Banken Deutschlands) | |
| 11:05 | Coffee Break | | | |
| 11:25 | **„Kennzahlenbasierte Tests im Data Warehouse“**<br>Adalbert Thomalla & Stefan Platz (CGI) | **“Agile Testing in Large Enterprise Projects”**<br>Sergey Zabaluev (C.T.Co) | **„Easy Trustee Modell”**<br>Thomas Hoppe (immofori AG) | |
| 12:20 | **Keynote:**<br>**“Illusions about software testing”**<br>Hans Schaefer | | | |
| 13:20 | Lunch | | | |
| 14:35 | **“Test Automation beyond GUI Testing”**<br>Hartwig Schwier & Patrick Jacobs (Océ) | **“Agile on huge banking mainframe legacy systems. Is it possible?”**<br>Christan Bendix Kjaer Hansen (Danske Bank) | **„Gesetzgebung! Wie kann ich mich darauf einstellen? Eine Irrfahrt für Kreditinstitute?“**<br>Marc Ahlbach | |
| 15:25 | Coffee Break | | | |
| 15:35 | **“The Data Dilemma”**<br>Huw Price (Grid-Tools Ltd.) & Edwin van Vliet (ABN Amro Bank) | **“Make or Buy?”**<br>Michael T. Pilawa (Pilawa SA) | **„Rechte der Supranationalen Aufsicht – Was müssen Kreditinstitute in Zukunft beachten“** | |
| 16:25 | Closing Session<br>José Díaz | | | |

**May 9–10, 2011 in Bad Homburg v. d. H. (near Frankfurt), Germany**

The two-day conference Testing & Finance, which is held once a year, brings together quality assurance specialists from the financial world from both home and abroad.

Please visit our website for the current program.

**Exhibitors**

Visual Studio · hp · CGI · steria mummert consulting · savignano SOFTWARE SOLUTIONS · macros Consult GmbH · VNB SERVICE · Díaz Hilterscheid · te testing experience

**Testing & Finance Europe 2011 –**
**A Díaz & Hilterscheid Conference**

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

info@testingfinance.com
www.testingfinance.com

www.xing.com/net/testingfinance

# Special challenges for testing in the banking industry

*by Dr. Hans-Joachim Lotzer*



This article highlights the special challenges of testing in the banking industry and discusses the approaches how these challenges can be overcome. Some of these challenges you will also face in other industries. However, the combination and the extent of these challenges is considered to be unique for the banking industry. Of course, besides the specific aspects discussed here there are many common testing challenges that are less industry specific and are therefore out of scope of this article.

## Characteristics of the banking industry with impact on testing

What is special about testing in the banking industry? There are *3 characteristics* that pose specific challenges to testing for banks:

1. Frequently changing market and regulatory requirements
2. High data confidentiality requirements
3. Complex system landscapes including legacy systems

What does this mean for testing of banking applications?

## Resulting challenges

### *Frequently changing market and regulatory requirements*

Strong competition in the market through products and services raises new functional requirements for banking applications in an ongoing manner. Besides this, there are frequently new or changing compulsory legal and regulatory requirements that need to be fulfilled and must be covered by the banking applications. This results in frequent releases and upgrades of business applications that need to be tested multiple times a year. Banks with individual software are affected as well as banks using standard solutions or application services.

Test execution for a single release typically includes an 80% portion just for regression testing – something that is not commonly seen as very exciting among business users besides the fact that this is not what business users are employed for. Whatever the involvement of business employees, IT employees or service providers is, running these tests multiple times a year can be very costly. Thus we face the first main challenge for testing in the banking industry:

*Challenge 1: Ensure cost-effective regression testing over the application life-cycle*

### *High data confidentiality requirements*

There are many good reasons to run tests with data copied from production systems.

However, it is not only the more or less strict data protection and bank secrecy law of different countries that poses special requirements to the use of production data as test data. It is also the risk of reputational damage, such as the loss of customer trust, that raises the level of data confidentiality requirements higher than in most other industries.

Having internal or external IT developers and testers work with production data for test purposes significantly increases the risk of law breaks, violations of need-to-know principles and reputational damage. This is not only because the number of users with access to production data increases, but also because the level of access control for test and development environments is not the same as for production environments.

The data confidentiality requirements rise with the strictness of the country specific laws and with the international distribution of business and IT. Thus the next challenge is:

*Challenge 2: Ensure test data usage is compliant with data confidentiality requirements*

### *Complex system landscapes including legacy systems*

The system landscape of banks are often extremely complex. The know-how about all the involved systems is scattered about the organization or even outside the organization. Typically we find many upstream/downstream data dependencies implemented as batch or online interface. These may include many cross application integrity constraints that - due to their complexity - imply the risk of being overseen when new releases are developed. The whole thing gets even more challenging when legacy systems are involved that have a far lower level of documentation and a much less clean data design than we would expect from modern applications. End of day batch processes in themselves are often extremely complex and can fill wall papers once we try to depict

them. In addition to this, there are many interfaces to external applications for market data and for settlement of transactions in the trading and payment area.

A consequence is that most data does not originate from the application where it is used, but from applications far up the stream. This makes it hard for testers or developers to produce test data that fulfills all those cross application data constraints. Thus our third challenge is:

*Challenge 3: Ensure testing covers system integration and ensure integrity of test data*

## Approaches to meet the challenges

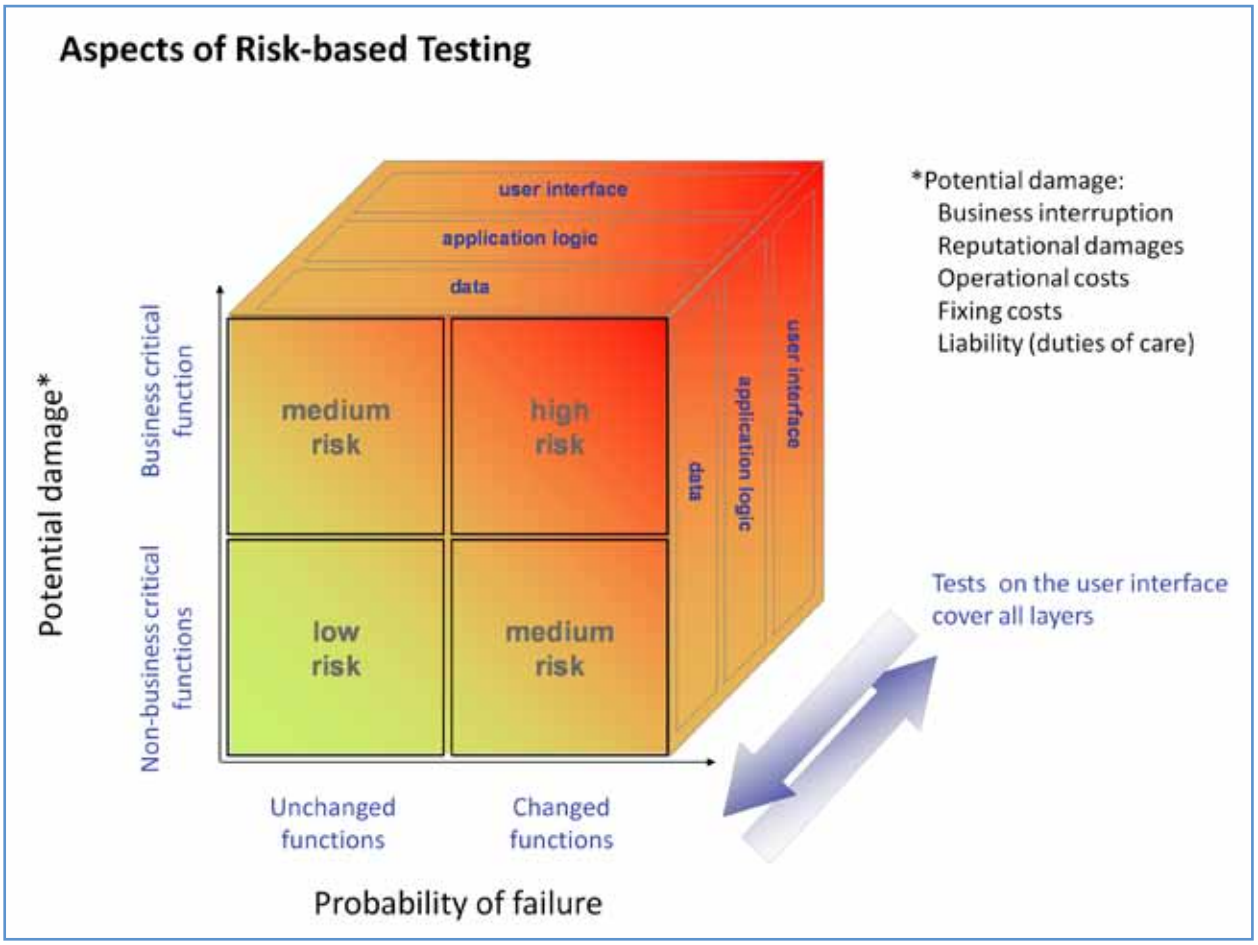
### *Ensure cost-effective regression testing over the application life-cycle*

Essential for regression testing over the application's life-cycle is that the test suite is managed across the releases, i.e., test suites should be maintained across releases, put under version control, linked to requirements, prioritized and test cases need to be precise regarding preconditions such as test data expected to be available in the test environment. Based on this the following approaches can be taken (single or combination) to achieve cost-effectiveness:

- Risk-based testing
- Automation
- Off-shoring

It is always a good approach to think about risk-based testing before starting to automate test cases or starting to off-shore test activities. The aim should be to reduce the number of tests to be executed for each release (or upgrade) while keeping the risk within an acceptable range.

It is the nature of regression tests that the rate of defects per executed test is significantly lower than it is for new functionality. Beyond that, the majority of defects found by regression testing – i.e., functionalities that worked fine in the previous release but now do not work anymore – are integration issues – i.e., side-effects of changes in one component or system that compromise functions of other components or systems. Consequently regression testing needs to cover higher integration test levels (i.e. end-to-end tests and tests on the user interface) rather than lower test levels (single applications/units or layers underneath the presentation layer/user interface). The well-known cost benefits of test execution on lower test levels through early defect detection are much less significant for regression testing than they are for testing of new functionality. Thus, in order to achieve cost-effectiveness, the execution of regression testing on lower test levels should be reduced to a technical verification, whereas the functional aspects should be shifted to higher integration levels. However, regression testing on lower test levels can make sense in special situations, when there are combinatory numbers of tests to be executed, which would mean higher effort when tested on the user interface (or end-to-end) level.



Test automation is something that can help to achieve cost-effectiveness of regression testing. The crucial success factor is sustainability, i.e. the automated test suites can be run in follow-up releases without too much maintenance effort. In order to have a good coverage with a minimal set of tests, the evident approach is to automate tests on a higher level – as discussed before under the risk-based testing aspects.

Off-shoring – i.e., relocation of testing activities or processes to countries with lower production costs - is another approach that should be combined with a risk-based approach and test automation. To be successful, off-shoring requires a pretty good readiness of testing regarding test suites, test data, documentation and processes. Furthermore a huge obstacle is challenge 2 "Ensure test data usage is compliant with data confidentiality requirements". When doing off-shoring the efforts towards this challenge rise significantly in order to be compliant with all cross-border aspects of doing testing from an off-shore country.

## Ensure test data usage is compliant with data confidentiality requirements

There are two basic approaches that can be taken to achieve compliance with data confidentiality requirements:

- Data masking
- Synthetic test data

Data masking tools and techniques can be used to change production data in a way that it is not sensitive anymore and can be used for testing purposes. The main advantages of this approach are a good coverage regarding diversity and volume of data and the option to reproduce production scenarios. For regression testing purposes it is a disadvantage that the underlying production data can change over time and lead to different test results (e.g. a test case using the same customer identifier at different times may have different results because the attributes of the customer have changed in the meantime). Avoiding this always requires additional effort to verify the test data or find appropriate test data. Regarding integrity it is possible to ensure the referential integrity even across different applications. But more complex integrity constraints (e.g. constraints based on mathematical calculations across different databases) can often not be ensured. The consequence is that not all data can be masked and that some compliance gaps remain.

On the other hand, the approach to generate synthetic test data is excellent under the aspects of compliance, sustainability and integrity – assuming that the synthetic data is produced the same way as real data in production. However, the disadvantage is that some gaps regarding coverage always remain, because it is not possible to synthetically produce all kinds and varieties of data (incl. historical data constellations) with all systems. In a less complex environment or, if integrity across systems is not required (e.g. for unit testing purposes), the coverage of synthetic data can be much better.

| | Data Masking | Synthetic Data |
|---|---|---|
| **Coverage** | ++<br>(diversity, volume, reproduce production issues) | -<br>(in complex system landscape there are always some gaps) |
| **Integrity** | +<br>(data masking might lead to violations of integrity) | ++<br>(when produced the same way as in production) |
| **Compliance** | +<br>(risk of gaps, some confidentiality requirements cannot be fulfilled) | ++<br>(all data confidentiality requirements can be fulfilled) |
| **Sustainability** | -<br>(underlying production data can change) | ++<br>(fixed data to base the regression tests on) |

## Ensure testing covers system integration and ensure integrity of test data

In order to ensure that testing covers the integration of different systems and to ensure the integrity of test data, there are not many alternatives to providing a system integration test environment (incl. all systems and batch processing).

The use of a user acceptance test or pre-production environment with production data for system integration testing is more a workaround than a real alternative. The problems with this are that a lot of IT staff need to work with real production data (resulting in higher risk regarding data confidentiality requirements), that the environment drifts away from being "production like", and that long UAT periods often result that put deployment dates at risk.

To serve its purpose, a dedicated system integration test environment needs to fulfill the following requirements:

| Requirements for a System Integration Test Environment | |
|---|---|
| **System Coverage** | i.e. all systems involved in the processing chain (up-stream/downstream) |
| **Operational Coverage** | i.e. running end of day processing, batches etc., ability of end-to-end order processing |
| **Data Coverage & Integrity** | i.e. coverage of most data constellations required for testing while ensuring data integrity |
| **Environment Management** | i.e. test coordination (test activities, deployments, data migrations), help desk, availability management, refresh of data & code |

A system integration test environment needs to cover the integration of all systems that directly or indirectly interact with each other (i.e. through processing chains of applications or just through logical data dependencies).. Due to the complexity of the system landscape, even the most mature software engineering processes won't eliminate the risk of overseen dependencies that might lead to system crashes, unwanted behaviors or disrupt data in production, if not uncovered by system integration testing.

The operational coverage of the system integration test environment ensures that all production batch jobs are also available for testing. Such batch jobs propagate data between different systems, or are required to process orders, complete transactions or perform closing operations.

The easiest way to achieve data coverage and integrity in the context of complex system integration test environments is to use production data copies. However, since there are also high data confidentiality requirements to be fulfilled, it is not that easy. In order to balance the data coverage/integrity requirement and the data confidentiality requirement, a systematic test data management function or service needs to be established.

Last but not least the importance of managing the environment increases with the complexity of the environment and number of people working with the environment.

## Conclusion

Things would be much easier if we could separately overcome one challenge after the other. Unfortunately, there are some conflicts to be aware of – i.e., there are approaches that serve to overcome one challenge but might simultaneously increase other challenges: Off-shoring makes it harder to fulfill the data confidentiality requirements, data anonymization might violate data integrity (especially when numeric values need to be anonymized), synthetic data generation often means concessions regarding coverage of system integration tests or concessions regarding data integrity when generated bottom-up (i.e. directly in the database) instead of top-down (i.e. via the same interfaces as used to generate data in production, and thus ensuring compliance with all business rules). In addition, the more complex the whole system landscape is, the more important it is that regression testing covers end-to-end integration and the more challenging it is to achieve cost-effectiveness.

Luckily, however, there are also some synergies: Synthetic test data can contribute to cost-efficient regression testing by ensuring that test data is defined, available and reproducible for all test cycles, which is an important readiness criteria for test automation and off-shoring. When generated analog to production data, generating synthetic test data can also foster data integrity.

No matter which approaches are chosen to overcome the special challenges of testing in the banking industry, the basic prerequisites to be successful are: test suite management, test environment management and test data management.

### > biography

**Dr. Hans-Joachim Lotzer**
*is Head of Testing Consulting at COMIT – a subsidiary of Swisscom IT Services focusing on IT Services for the financial industry. He has more than 20 years of professional IT experience – many of it in the area of testing and quality assurance. Other areas include process engineering, project management, business development, requirement engineering and software development. He studied mathematics with computer science at the TU Darmstadt and received his Ph.D. in business informatics at the University of Mannheim. His current fields of interest are managed testing services, test data management, testing maturity & KPIs as well as risk & compliance topics. Feedback is welcome at hans-joachim.lotzer@comit.ch.*


# Data and application migration

**What knowledge and skills does a test manager need?**

*by Iris Pinkster - O'Riordain*

The theme of this edition of the Testing Experience is "Testing @ Domains". In this article the author looks at the domain of data and application migration. A lot of the material discussed here is based on the author's experience of working in large and complex migration programs.

More and more companies face data and application migrations. This also implies that in our job as testers and test managers we will run more and more into migration projects. The business drivers can be quite diverse. They can range from mergers, acquisitions and business process outsourcing to more technical reasons like the employment of a new operating system or server / storage technology replacement or consolidation. Other examples include: virtualization, relocation of a complete data center, server or storage equipment maintenance and workload balancing.

A few statistics. Bloor Research found in a survey in 2007 among Global 2000 companies that the market for data migration exceeds US$5 billion, 80% of migration projects ran out of time or budget and 20% of the projects have no separate budget or timescale. Hitachi measured in 2010 that migration project expenditures are on average 200% of the acquisition cost of enterprise storage, enterprise storage migration costs can exceed US$15,000 per terabyte migrated, and in storage migration projects 70% of time is used for planning, 30% for executing.

These statistics show that many improvements can be made in this huge market. What domain knowledge should testers and test managers have, to cope with these migration projects from a testing perspective? This article will give you some more in-depth information. The author will lead you from a general overview on the various types of data and application migration to more specific consequences for test strategies to test these migrations.

## A migration is not just a migration

Whatever the business is you are working in, when faced with a migration you have to fully understand what type of migration you have to test.

When testing a data migration, it is necessary to understand what the target system is. It will hardly ever occur that the target environment is exactly the same as the original environment. Differences can occur in type of database, version of database or in the migration of the information to a complete new environment. The bigger the differences, the harder it is to validate the data after the migration. A simple reconciliation[1] might not be possible.

When testing an application migration, you have to understand the migration strategy chosen. If the migration is executed via a binary copy, the test effort and emphasis will be different from when virtualization is chosen. And everything will get even more complicated when an application migration is combined with an upgrade to a newer version, or if data pollution within the application is handled at the same time.

In the next section we will look in greater detail into various possible migration strategies and the consequences they might have on the test strategy to choose.

## Migration strategies and the consequence for the test strategy

For data and application migration various migration strategies exist. In this section five different types will be explained with

*1 Reconciliation is the action of performing a check that a certain application or object is migrated successfully where no data is lost during transfer. For example, if a database is copied to another environment, a reconciliation approach could be the check if all the database files are transferred correctly by calculating a checksum on each file. However, if a database upgrade is performed during the move, a checksum approach will not work (you will get different checksums). For an upgrade, a more detailed reconciliation approach is necessary, where all the objects inside the database must be checked (tables, records, constraints, etc).*

their potential consequences for the chosen test strategy. Note that this selection is not exhaustive. When reading this section, you will get a better understanding of the importance of knowing what type of migration is used within your specific projects.

### *Rebuild and data migration*

The migration strategy 'rebuild and data migration' recreates the existing application environment, which may consist of several IT objects (such as databases, interfaces, software). Once the newly created application environment is recreated or 'rebuilt', the production data from the existing application environment can be copied to make the new environment - the production environment. Ideally this has to be done in the common steps via the environments: Development, Test, Acceptance and Production[2].

**Impact on testing**: full extensive testing is necessary. An application is completely rebuilt with all the possible errors that come with building something new. Before the new situation is implemented, testing has to be done to make sure that the business is not harmed by the migration. So besides functional testing, also connectivity testing, performance testing and business continuity testing have to be considered. On top of that, a data migration test has to be performed when the database is completely rebuilt. A trial run with production data should ensure that all data is migrated correctly and the migration of the data fits the migration window.

### *Binary copy*

The migration strategy 'binary copy' involves creating a snapshot image of an IT object and reinstating that IT object in the new environment. This involves 'bringing down' the IT object (the database, application server, etc.) so that all changes are committed to the data store before a binary copy is created. The exact procedure of performing a binary copy differs per IT object and must be specified explicitly.

**Impact on testing**: as an exact copy of an application is made in the new environment, a full test cycle might not be necessary. To check whether the migration is successful, a simple intake test in the Development and Test environment (when applicable) might be suitable. Whether more in depth and regression testing on the Acceptance environment is required depends on the risk or business criticality of the application. After a binary copy, a test of the business processes is necessary to check whether the customer is not affected by the move. This is even more crucial in the case of many interfaces that have to be adjusted to work with the new environment.

### *Lift and shift*

The migration strategy 'lift and shift' involves the physical move of servers from one environment to the other. This method involves bringing down the servers, disconnecting them and deploying them into the new network. All applications that are running on the server are deployed in the new environment as well.

As the 'lift and shift' strategy must be performed using a big-bang

*2 These environments will be referred to as D, T, A and P respectively in the rest of the article.*

approach[3], the risk of business discontinuity is high, due to the fact that no real fallback scenario is in place after the production hardware is disconnected. Therefore, the risk of technical failure is high as well. Especially old hardware might fail because of the physical transportation. There is a substantial chance that it will not start up after the move.

**Impact on testing**: which tests are necessary depends on the scope of the 'lift and shift' hardware. No interface testing is necessary for parts that are within the same 'lift and shift' environment. Depending on the outcome of the product risk analysis, testing with stubs and drivers to test the interfaces with applications outside the 'lift and shift' environment is necessary. Regression testing in Acceptance depends on business criticality and the size of 'lift and shift'.

Testing of processes doesn't have to be executed when the process exists fully within the 'lift and shift' environment.

### *Change of IP address*

The changing of an IP address is in general not a migration strategy. Instead, this option illustrates that some applications are already running on the intended hardware in the correct data center and therefore do not have to be moved. However, one final step is required to complete the migration, being the changing of the IP address. Other systems will connect to the new system after this IP address change.

**Impact on testing**: the risks of changing the IP address of a server on which applications are running should be minimal when tested correctly; mainly because the effort of changing an IP address should not require much effort and a fallback scenario is easily achieved (change it back). Therefore, the risk of business discontinuity is low and the risk of data loss as well (relevant queues in middleware should be emptied before the IP address is changed). Interface testing is the most important part of testing. The scale of acceptance testing (regression) depends on the business risk.

### *Virtualization*

The migration strategy 'virtualization' for server applications involves the creation of an image of the server that will be moved to a virtual environment in a data center. Often this is combined with a change of IP address as mentioned above. Another possibility is virtualizing the client and/or local installed applications. In that case the applications have to be scripted (packaged) in order to work in a virtual environment in a data center. That way users can use their applications running in the data center on their local machine (that now could be a thin client). The screens of the applications are "streamed" to their local machine.

**Impact on testing**: for most server applications, the same testing

*3 The migration strategies can be implemented following a big-bang migration, an incremental migration, a bi-directional synchronization or a combination of those. This also impacts testing.*

as with 'lift and shift' is required. For server applications which require very high processing power (e.g. index and search engines for document management systems) extra performance testing should be applied. The performance penalty of using a virtual environment in such a case can have a big influence on the performance of the system. For client applications extensive regression testing is necessary. Scripting client applications to work in e.g. a Citrix environment might have a huge influence on the functional behavior of the application. Besides that, not all client applications are suitable to run in a virtual environment. There are also application suppliers that do not support their applications when they are implemented in a virtual environment. The scale of acceptance testing (regression) depends on the business risk.

## Consequences on (test) environments

For all testing activities mentioned in the previous section, one of the most important entities is the test environment. This environment is constantly changing and it is crucial to keep a good understanding of what applications and data are implemented on which environment, especially when changing applications and the corresponding data to a complete new data center.

Here an example. In figure 1 you can see the situation that one business process follows four applications (A, B, C and D) in one data center. The input could be a manual action like a customer ringing the helpdesk. This will start the business process. The output could be a letter or email to the customer.

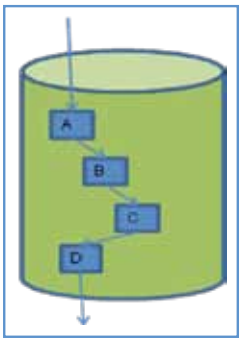



Figure 1: All applications and data in one data center.

In an incremental migration scenario, B is the first application that will be moved to a new data center. Figure 2 shows this new situation.

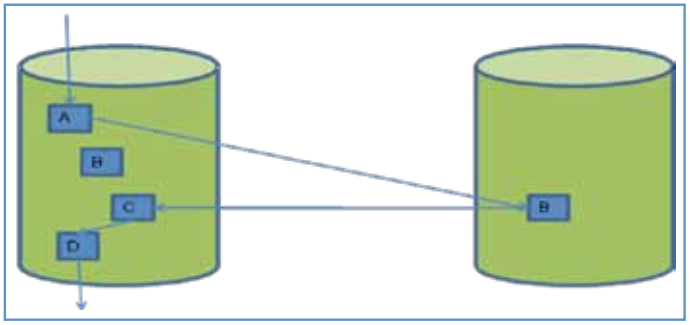



Figure 2: After an incremental migration application B exists in the new data center.

Before this situation will be implemented it also has to be present in the test environments. In this situation extra emphasis has to be put on the new interfaces between A and B and B and C. Can the applications reach each other? Is there a firewall between the data centers?

Has the temporary "longer" line a negative influence on the performance of the business process? Also a test has to be done to ensure that the old connections between A and B and B and C are disconnected.

Maintaining detailed drawings of the environment and preserving IT object version control are crucial during an incremental migration.

When a first application (or first set of applications) is moved, it is possible to test business continuity by executing disruptive test scenarios on the new environment (e.g. emergency power downs). After moving the second application set, executing disruptive test scenarios might not be possible as they could negatively impact your business processes in production. The possibilities and threats are only known to you when you have a detailed and up-to-date technical architecture.

## Migration challenges

In the previous sections, different situations are described that influence testing. When working on data and application migration the following points also have to be taken into consideration before commencing the project:

- Minimize disruption to business processes;
- Ensure all data is migrated and accounted for;
- Handle the complexities of mapping data from various legacy systems to the new application(s);
- Handle existing data pollution;
- Handle customer orders already in progress.

## Conclusion

The necessary domain knowledge when you as a tester or test manager are involved in a migration project is not necessarily lying in the functional domain knowledge. What you need in these types of projects are people that understand the technical landscape and the consequences for testing that come with the chosen migration strategy.

After reading this article you will hopefully have a better understanding of the complexity and the domain knowledge necessary to deal with testing of migration projects.

### > biography

**Iris Pinkster - O'Riordain**
*is test advisor at Professional Testing and has experience in testing and test management since 1996. She co-developed Logica's method for structured testing: TestFrame®, their test management approach (Risk and Requirement Based Testing) and TestGrip, the method on test policy and test organization. She is co-author of the books published on these topics. She often speaks at (inter) national conferences. In 2007 she won the EuroSTAR award for "Best Tutorial".*

# Testing @ Domains, and the winner is .....

Column

*by Erik van Veenendaal*

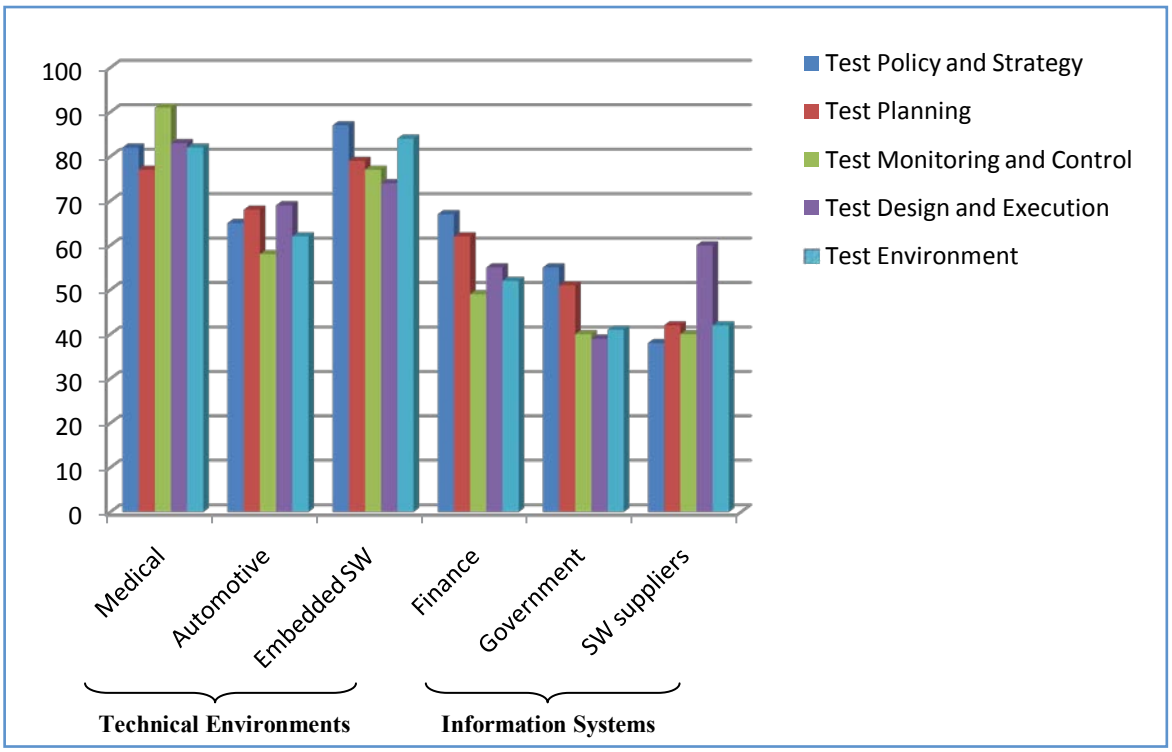
*"Risk is the possibility of an undesired outcome. Risks exist whenever some problem may occur which would decrease customer, user, participant, or stakeholder perceptions of product quality or project success. Risk is often perceived as a value that would vary depending on the circumstance or the perspective. Risk is what is taken when balancing the likelihood of an event vs. the impact if it actually happens. In effect what are we willing to leave to chance? In most cases the level of risk the business is willing to take is dependent on the amount of time and available budget. At a closer look risk-based testing is highly related to the concept of good enough testing. Good enough testing reflects what we do in real life where something less than a "perfect" solution is inevitable. The good enough paradigm is helpful to understanding the risk-based test approach. It provides a mental framework for the (release) decision-making in projects where risks are being taken."*

I started this column with an excerpt from an upcoming book on product risk management since the differences in testing between in the various domains should be driven by the different levels of risk of the systems being developed in the various domains. We can discuss the differences for hours, days or weeks, but testing is performed for a reason. It is performed to mitigate product risks. A system that has safety risks (e.g., a medical system) should be tested more thoroughly than a system that "only" has financial risks (e.g., a banking system); a system that only has financial risks should be tested more thoroughly than an internal logistics system etc. Although we all agree with these statements, is this what it looks like in the real world as well? Even when a medical system is being tested, there is pressure to deliver on-time and beat the competition. Will a business manager at an insurance company accept less thorough testing (compared to a medical system) since there is only a financial risk?

With over 25 years of practical experiences in the software industry, I have worked in many different organizations in various domains. I have also run many courses over the years, both public and in-house, and have discussed how testing is performed in their organization with a many participants. I too believe there are many differences for many reasons, the risk level of the system under test being probably the most important one. Do we have data to substantiate our subjective feeling?

### *Knowledge and skills of the test professional*

Over the years I have kept track of the pass rates on ISTQB Advanced and ISEB Practitioner. Out of interest I organized the data




Graph 1: Test maturity in the domains

such that it allows for a distinction between test professionals working in a more technical environment, where usually multi-disciplinary products are being developed (e.g., automotive, medical, embedded software, mobile), and those working in a more office type environment where database oriented information systems are being developed (e.g., banking, insurance, government, trading). Surprisingly (or not?) the pass rate for test professionals working in a more technical environment comes out almost 20% higher! Having worked in both types of environment, I'm not too surprised, but it does make you think .....

### *Tools uptake*

Recently, I published a paper on the results of a tool survey [1] in which the same distinction between technical environments and database oriented information systems is made. A general tendency in tool uptake is that in the area of technical environments substantially more tools are available and applied than in the area of information systems. This is true for most types of tool. Some striking examples are shown in the table.

| Test tools implementation ratio[1] | Technical Environments | Information Systems |
|---|---|---|
| Requirements management | 29% | 9% |
| Configuration management | 76% | 39% |
| Static analysis | 47% | 12% |
| Coverage measurement tools | 21% | 4% |
| Dynamic analysis tooling | 34% | 7% |
| Performance tools (incl. load/ stress) | 40% | 27% |

Table 1: Tool uptake in the domains

For some tool types the fact that the uptake is much lower for information systems can be explained by the fact that fewer tools are available for the languages they are using. The difference in tool uptake perhaps also relates to a more professional way of doing software engineering and component testing in technical environments. The differences in uptake with tools such as dynamic analysis, coverage measurement and static analysis seems to be an indicator for this. However, doing a detailed analysis is not the objective here. Again, a substantial difference between the domains, it does make you think .....

### *Test maturity*

Finally I have tried to gather data on the average test maturity in the domains using the TMMi framework as a reference model. TMMi is rapidly becoming the world-wide standard for measuring and benchmarking test maturity. Together with some friendly organizations that also perform formal and informal assessments, I managed to get the data from several dozens of TMMi assessments. The graph hereafter shows the average ratings (scale 0 – 100) for some domains in the TMMi maturity level 2 process areas.

A graph that is very interesting to analyze and discuss more in detail. But again, we notice a substantial difference between the technical environments and the information system environments. In my view, the product risks (e.g., safety, reliability, and cost of recall) of the medical, automotive and embedded software industry drive the organizations towards a more thorough test process. Note that this does imply that agile software development and exploratory testing and the like are not practiced in technical environments. On the contrary, and when practiced, they even seem to be more successful!

### *Finally...*

What does all of this mean? "To measure is to know." It is good to have an awareness on the differences that were discussed. From various perspectives it seems some domains clearly have a higher test capability. This is reassuring, since I would like my car to stop when I use the brake, my TV set not to reboot during a soccer

1 *Implementation ratio is defined as the number of organizations using a certain test tool divided by the total number surveyed.*

match and, of course, to receive the right radiation doses when being medically examined. If you want to improve your testing, benchmarking is often a good idea; however, you may want to have a look at domains that are in the premier league of testing. Some of their fully deployed practices could also be beneficial to you. When someone presents a new idea at a testing conference and presents the results based on an internet application or in the Microsoft world, I'm not always fully convinced. Please also show me how it performs in a more advanced, challenging and critical environment!

Note that the column is about testing in the various domains in general. There are many organizations that will perform a lot better or (sadly) a lot worse than the domain average. Where are you?

[1] E. Van Veenendaal, Tools and the last six years, in: Professional Tester, November 2010

## > biography

*Erik van Veenendaal (www.erikvanveenendaal) nl) is a leading international consultant and trainer, and a widely recognized expert in the area of software testing and quality management with over 20 years of practical testing experiences. He is the founder of Improve Quality Services BV (www.improveqs.nl). 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 various domains for more than 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, vice-president of the International Software Testing Qualifications Board (2005–2009) and currently vice chair of the TMMi Foundation.*

*First edition | June 23rd, 2011*

# test automation day

www.testautomationday.nl

## June 23rd 2011, Stadion Galgenwaard, Utrecht - The Netherlands

***Today, 'Test automation' is a much talked about topic in the world of software testing and quality. The next generation test tools facilitates a faster, better and cost-effective test process. So there is much to gain, but only if we use the right tool in the right place.***

This challenge in the "World of Testing" has led to the first edition of the annual Conference for test automation practitioners and experts on June 23rd:

**Test Automation Day 2011!**

The program consists of (inter)national keynote speakers, best practices and workshops. In collaboration with an independent Program Committee, CKC Seminars ensures an interesting day with lots of Test Automation content.
The speakers will contribute to the main theme:
*"Optimizing the profits of the next generation Test Tools."*

One of the top experts presenting on the Test Automation Day 2011 will be Elfriede Dustin, Software Engineer at Innovative Defense Technologies (USA) and author of the book 'Implementing Automated Software Testing: *How to Save Time and Lower Costs While Raising Quality' (2009).*

**In short: This is the 2011 Test Event not to be missed!**

**Register before April 1st, and receive an early bird discount of €100,- excl VAT.**

**A unique chance to visit this conference for only €195,- excl VAT!**

## Register now at www.testautomationday.nl

*Conference organization and initiative CKC Seminars | Founding partner Squerist*

ckcseminars SQUERIST

# New experiences in testing component based automotive systems

*by Muzammil Shahbaz & Robert Eschbach*



Road vehicles no longer consist of just hardware and mechanical parts. They now also contain millions of lines of code in various embedded parts: from power train to infotainment devices. The modern intelligent systems such as automated parking, adaptive cruise control and telematics, demand a rigorous software engineering discipline in the automotive industry.

In this article we shall discuss the impact of this evolving paradigm on the traditional process of automotive engineering. In fact, the usual manufacturing process is *modular* – as can be referred in software engineering terms. The components of cars are produced by third-party suppliers and OEMs concentrate more on the assembly of components. A large degree of the production process lies in outsourcing, parallel development and system integration. Now, the software engineering of automotive components has also been distributed through the network of n-tier suppliers. All top-notch automakers have already put effort into building the architecture for *plug-and-play* of these components through dedicated frameworks, e.g., *iDrive* from BMW, *OnStar* from General Motors, *Sync* from Ford, to name a few. Keeping this landscape in view, we provide an overview of emerging problems in the automotive industry and then tell a story of our recent experience in testing an embedded automotive system that was composed of third-party components.

## Problems and remedies

The automotive industry experiences imprecise specifications for the ready-made software components that are usually procured from different suppliers. According to our experience, the four major aspects that largely contribute to this problem are:

- Non-synchronization between OEMs and suppliers on software engineering process
- Limited transfer of intellectual properties
- Staggering number of component variants
- Malpractices of specification maintenance

This is an undesirable situation in an industry where quality and safety features have uncompromising priority. Because of the multi-tier environment, the situation is tackled in a very ad-hoc manner. Not every component is provided with a complete specification document, design details or source code. In most cases, engineers use domain expertise and/or carry out extensive simulations to reveal additional information about the "black-box" components. Since the specification is usually incomplete and imprecise, the methodologies for testing, validation and verification are restricted only to the interface levels of the integrated systems. In order to understand the complete interactions between the components, understanding only the syntax at the interface level is not sufficient. Instead, an understanding of the complete input and output behavior at the functionality level is required.

Unfortunately, there is no effective solution for this emerging problem. In general, it is a matter of experience and in-depth knowledge of the underlying features of the automotive components that help in revealing additional information. The more the engineer is experienced, the more understanding is obtained within the optimum time. A systematic and comprehensive treatment is, however, missing. The usual approach is running the system over a few test scenarios and then observing the corresponding behaviors. After gaining initial understanding, the observed behaviors are recorded in a textual or some semi-formal model. The model then serves as an input to comprehensive testing methodologies. The execution of test cases produces additional results that help in fine-tuning the initial observations and thus provide more understanding. With more experience, engineers build their in-house techniques to run integration over third-party components. Any formal technique in this regard is just another facet of a reverse engineering process.

## Reverse engineering in action

We stumbled upon such a situation in a recent project of testing embedded systems of an advanced car from a large OEM. The area of focus was the integrated door control system that was built up of third-party *Electronic Control Units (ECUs)* (1). The system had a heterogeneous design but there were three principal ECUs that made up the system. They were *locking system, power window (WCU)* and *mirror (MCU)*. Plus, it was equipped with subsystems such as telemetric and temperature sensors. A 60+-page textual specification was provided by the OEM describing the behavior of the ECUs with diagrams of the physical interfaces. Each one has its own set of requirements, which were apparently orthogonal to each other. The ECUs were collected from tier-1 suppliers

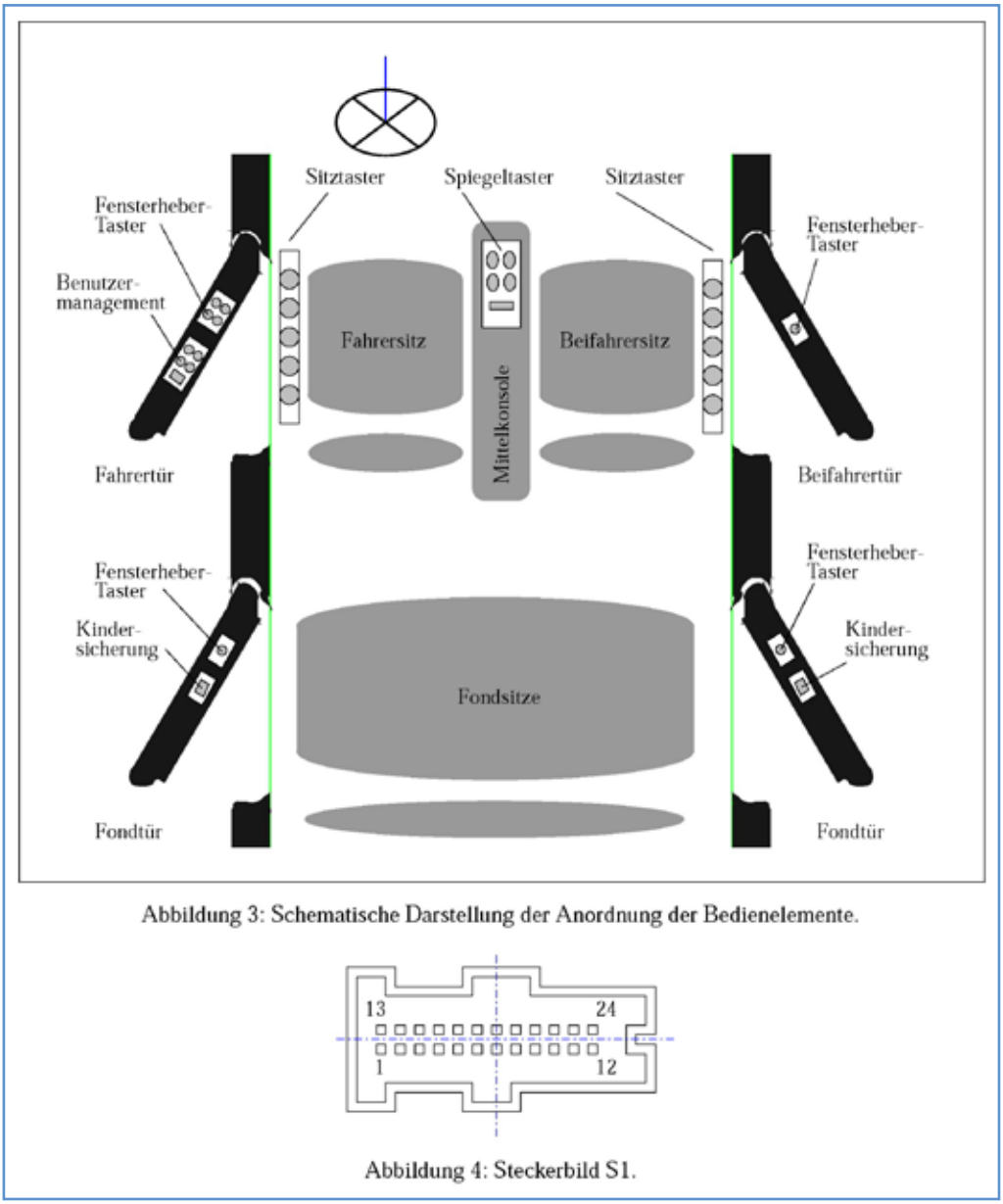



Figure 1 Snapshot of the specification document

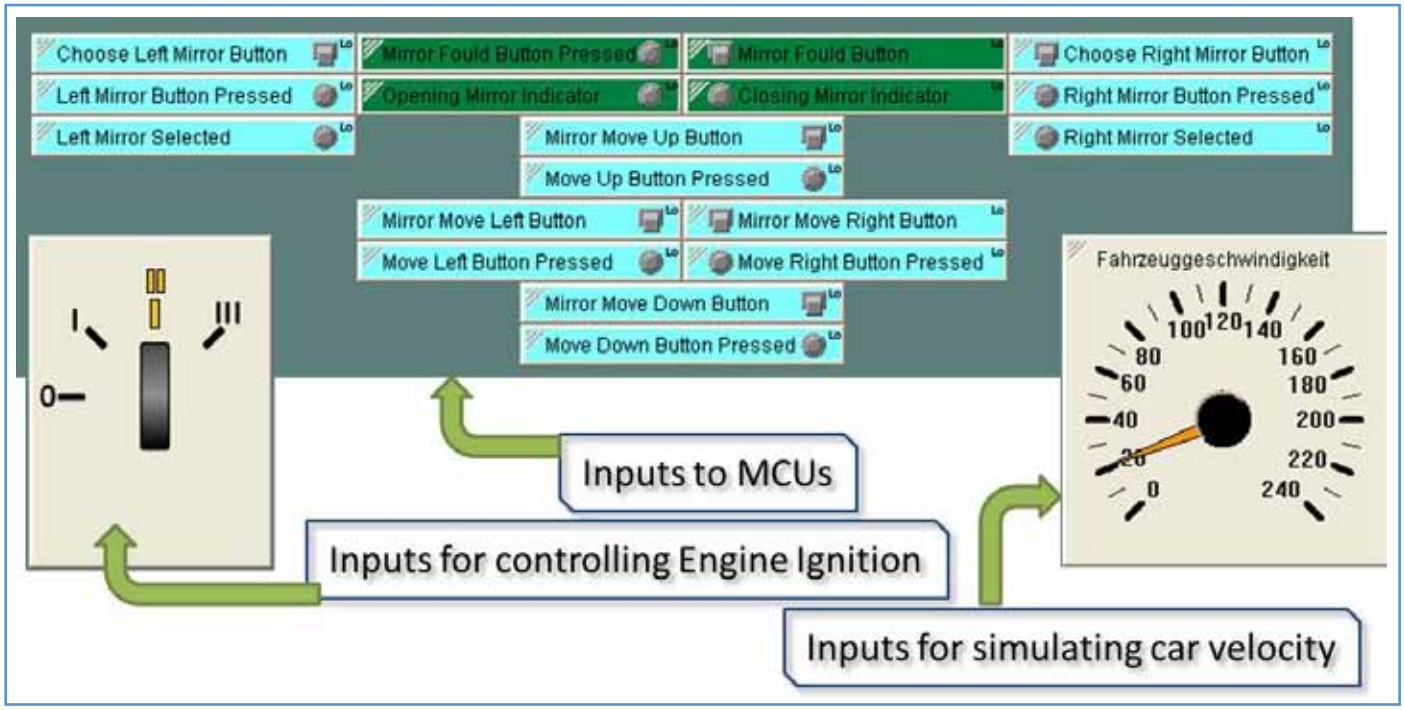



Figure 2 Signal Inputs for Door Control System

and integrated by the OEM into the final door control system. The specification of the system was by no means complete. The ECUs were black-boxes with no source code or design models given, but some general behaviors and few input/output signals were specified. According to the specification, an abstract and simplified view of the ECU behaviors is given as follows.

- **MCU** controls mirror movement around its horizontal and vertical axes and also folding and expansion of the mirror frame with specialized motors. It is also equipped with a heater that responds to the outside temperature and turns on/off heating automatically.
- **WCU** controls the window's opening/closing movement with the help of specialized motors. It is equipped with position sensors to stop the motor rotating when the window is opened/closed completely. An obstacle detector is also embedded that interrupts window closure when an obstacle is detected in the window frame.
- **Locking System** controls the automatic locking/unlocking of the door.

In the following, we describe the methodology of extracting unknown behaviors from the interactions of ECUs in the door control system.

## The methodology

The goal of the project was to test the integration of the door control system. We could have applied Hardware-in-the-Loop (HiL) (1) testing and put the system under closed loop. Since the system was composed of black-box ECUs, simulating ECUs for a few test scenarios would not be sufficient to find out the missing requirements, viz. the interaction graph of the ECUs. Here, we devised a systematic reverse engineering methodology to recover the formal behavior model of the ECUs.

The basic information about the interfaces was provided in the specification document with figures and some input values. A snapshot of this can be seen in Figure 1. However, this was not enough to drive a successful reverse engineering process, which usually requires a larger input set in order to reveal the hidden behaviors. Therefore, we extracted the additional inputs (or signals) that could stimulate the ECU functions from a well-known test-runner in the automotive industry: PROVEtech:TA. The tool is equipped with a GUI for monitoring sensors and actuators of the ECUs graphically (Figure 2). It also provides a common vocabulary of CAN/LIN bus (1) signals, so that a tester can observe the signal values while ECUs are active. We prepared a set of abstracted inputs for the door control system by selecting a few dozens of signals from the vocabulary.

The central point of the reverse engineering process was a learning algorithm (2) that is implemented in our in-house toolset. The tool takes the input set of the system and derives tests from it to stimulate the system by applying those tests. It then records the system observations in terms of execution traces (or I/O behaviors). The tests are applied iteratively until some criterion on the recorded observations is satisfied. Finally, the tool applies learning algorithms to conjecture a state machine model that depicts the behaviors of the system consistent with the recorded observations (see Figure 3 for an overview).

We provided the input set of the door control system to the tool and connected it with PROVEtech:TA for running the tests on the system. The real-time interfacing between the tools and the door control system was provided by a HiL simulator. Figure 4 shows the physical setup of the automated reverse engineering process.

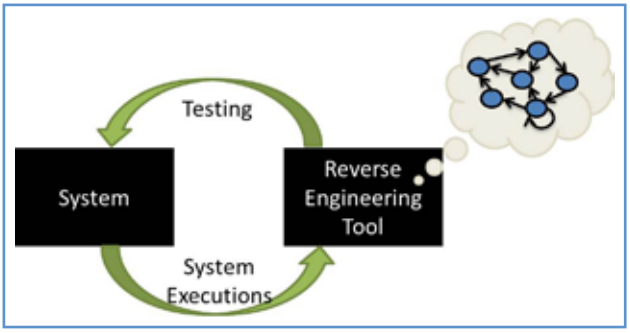



Figure 3 Reverse Engineering Methodology

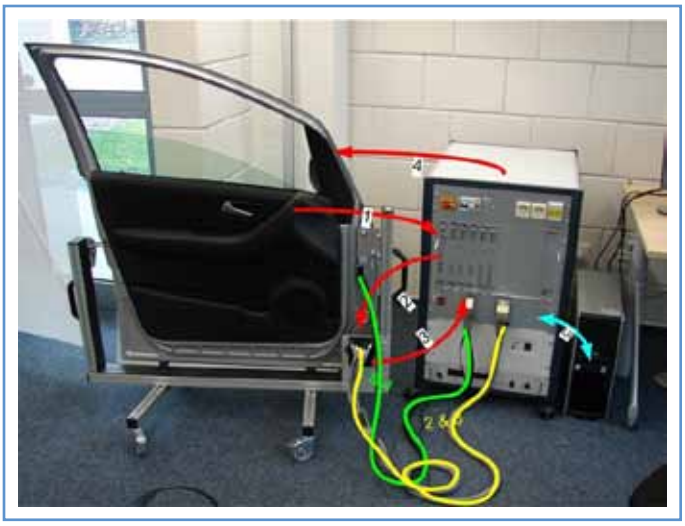


Figure 4 Test Setup

The reverse engineering process obtained state machine models of the behaviors of the ECUs in about eight hours of test execution. The largest model was obtained for the MCU that comprised of 29 states and 94 transitions. We found several interesting observations that ranged from minor to complex interactions between the ECUs. Nevertheless, the interactions were missing in the specification document, and thus gave added value to the integration testing. For example, MCU did not operate on the *expand mirror* command if the car velocity was beyond 50 MPH. Another behavior from WCU was that it did not close the window if preceded by *window opening command* in the span of 5 seconds. The state machine models had graphical illustrations of the behaviors revealed in the reverse engineering process and it was easy for a deep inspection.

The results achieved in this experiment raised confidence that the imprecise specification can be compensated by employing a reverse engineering methodology. In practice, however, this is not at all a smooth and easy-going process: there are snags in various steps that engineers have to tackle before it can be used. The most inconvenient steps are preparing a list of inputs for stimulating ECU behaviors in a specific context, mapping the abstract to concrete inputs (and vice versa) in the testing tools and realizing how much of the hidden behaviors are captured by the process, especially in the case of complete black-box ECUs. We believe that a proper toolset and reasonable experience can bring down these snags to the extent that the reverse engineering process can be feasible in practice.

## Conclusion

Road vehicles are built up of multiple third-party components that are collected from n-tier suppliers. The components use different architectures and even different communication protocols. Well-structured and detailed specifications are essential for quality integration. There are many techniques in the formal modeling world, like UML and state machines, for the behavioral description, and methodologies for maintaining specifications in the development lifecycle. The challenge is making a standard procedure across the OEM-Supplier chain that, unfortunately, seems quite difficult to attain with a decentralized community that is more and more diverse. New trends in the automotive industry are witnessing that the practices of testing and reverse engineering go hand-in-hand.

## References


1. **N. Navet, F. Simonot-Lion**. *Automotive Embedded Systems Handbook*. s.l. : CRC Press, 2008.

2. **M. Shahbaz, R. Eschbach**. Automatic Discovery of Unspecified Behaviors in Automotive Control Software. *TAIC PART*. 2010, S. 181-188.


## > biography

**Muzammil Shahbaz**
*is Senior Engineer at Fraunhofer IESE. His interests cover the practical applications of formal methods in embedded systems. He has been involved in testing and verification projects at Orange Labs, STMicroelectronics and Hewlett-Packard. He earned his PhD in Software Engineering from Grenoble Institute of Technology, France.*

**Robert Eschbach**
*is the department head of Embedded System Quality Assurance at Fraunhofer IESE. He is engaged in projects with safety-critical and software intensive systems, particularly in the automotive domain. He earned his PhD in Computer Sciences from University of Kaiserslautern, Germany.*


# The Woodcutter and The Tester

*by Robson Agapito Correa (Revised by Graziele de Queiros)*

There was a very strong woodcutter, that had high productivity in his work, but he was very snobbish and thought he already knew all about his job. If you compared him to other woodcutters, he had an advantage, because he was stronger.

However around the city there was a famous woodcutter who he was a myth in the woodcutter community. Although he was an older man, he was a champion to the woodcutter community.

One day, the younger woodcutter, in his arrogance, challenged the older woodcutter, and the older woodcutter accepted immediately. People talked about "The Challenge", neighboring cities were eager for the competition. And a lot of people went to watch the great event.

On the day of the event, a couple of woodcutters were waiting for the start, and after the buzzer sound, they start "the fight". The older woodcutter goes right and the younger goes left. The younger works hard and believes that he will be the winner because the older man doesn't have the same strength in the arms as he does. The younger woodcutter doesn't stop for a minute, and at times he looks at the older woodcutter and notices that the older man has stopped to rest. He thinks arrogantly: "I am already the winner of this competition".

In the evening, after they finished "The Challenge", when the results were calculated... Surprise! Everyone realized that the older man was the winner. The younger man was upset and he didn't know what had happened, he went to the older man and asked him: "What did I do wrong?". The winner answers him: "You believed only physical strength and willpower were enough to beat my knowledge; nevertheless we realized that it is not. While you looked me when I stopped you believed that I was resting, I didn't rest, I stopped to sharpen my axe."

As we can see from the parable describe above, it is important to use and to prepare a correct tool and we need to know what the best moment and the ideal calibration is for the tool to give the best performance.

For testing of software it´s the same; if you have a tool to measure performance and you don't know how to use it, or you don't know how to set up the environment, or if you don't know that there is high business risk connected with the system's performance, do not use performance tools. You have to connect with the software objective that you are testing and you have to know what the ideal configuration is for the test environment to be equivalent to the production environment.

When we bring together the younger woodcutter's willpower and the older woodcutter's expertise, we have an ideal professional for the competitive business market.

We always have to be an enthusiastic young woodcutter (or tester) and we will always have to search for more knowledge to gain expertise like the old woodcutter.

## > biography

**Robson Agapito**
*is coordinator of a test team at Mega Sistemas Corporativos in Itu, SP/Brazil. He has worked in software testing since February 2007. At Mega Sistema, he acquired knowledge and learns a lot every day.*
*He worked as developer for eight years, which helped him to see both sides (as tester and as developer). Today, he is able to negotiate with managers and directors, as well as with developers and testers.*
*He is under-graduated at Fatec (Technology College in Americana-SP) in data processing, graduated at PUC (Pontifica Universidade Católica in Campinas-SP) in System Analysis with enphasis on client/server, and graduated at Fatec (Technology College – in Sorocaba-SP) in Information Technology at Manufactoring.*
*He worked as a teacher at Fatec (Technology College) in Indaiatuba-SP and he taught Information System Planning. He teaches at Iterasys, a company specialized in quality assurance and software testing. He is certified by ALATS® (CBTS – Certified Brazilian Test Software), by ISTQB® (CTFL – Certified Tester Foundation Level) and by Scrum Alliance (CSM – Certified Scrum Master).*

# 1 month FREE access

**Testing Experience in association with *learn*testing brings you an ebook library comprising 50 of the best software testing books from many leading figures in the industry:**

### Subjects include

- ISTQB Certification – Foundation Level
- ISTQB Certification – Advanced Level
- Agile Development & Testing
- Performance Testing
- Test Automation
- Test Management
- Test Techniques
- Java Unit Testing
- Testing .net Applications
- Usability Testing
- Games Testing

### Authors include

- Dorothy Graham
- Lee Copeland
- Anne Mette Jonassen Hass
- Cem Kaner
- James Bach
- Bret Pettichord
- Boris Beizer
- Paul Jorgensen
- Isabel Evans
- Ilene Burnstein
- Bill Hetzel
- Erik van Veenendaal
- Rex Black
- William E. Perry

**Testing Experience** is delighted to offer its readers **1 month FREE access** to this wealth of literature, but hurry - there are only a limited number of subscriptions available.
**As an extra bonus**, you also have access to the new ***Learn**testing* 'Symposium' of videos, papers and presentations from thought leaders in the software testing industry.

**Visit www.learntesting.com/products/ebooks and subscribe for ebooks + Symposium using coupon T1MONTHFREE in the shop* to get free access**

*Select Euro currency in shop


# "Testing PetroVR, a software solution for the Oil & Gas Market" or "Testing in the Oil & Gas Market"

*by Leandro Caniglia, Damian Famiglietti y Ernesto Kiszkurno*

Technology is widely used in the oil & gas industry. Both software packages and custom developments share unique features and terminology that impact the way testing is done. Some of these characteristics are:

- **The leading role of data**. Software has to handle large amounts of data, normally processing a significant set of inputs into an even larger number of outputs which contain sensitive and valuable information. Data environments are highly complex and interconnected.
- **The complexity and extent of functional knowledge needed to operate it**. The software may contain complex calculations, make use of statistical models for simulations or projections, and interact with multiple systems that feed its processes with critical data.
- **Specificity of the technical vocabulary used**. This is seen both in the dialog with the different types of users and also in the software itself.

At the same time, if the software is to be installed at different clients and on a significant number of computers with heterogeneous environments, the complexity of the development and of the required quality levels increase.

## The product and the development process

PetroVR is a comprehensive *business simulation suite* for the upstream oil & gas industry. It integrates knowledge from different domains: a) *Engineering*: reservoirs, drilling, facilities, fluids, production and injection, routing and pipelines, resources, maintenance; b) *Economy*: capital and operational expenditures, depreciation; c) *Planning*: multi-decade scheduling of interdependent activities influenced by inflation, learning and uncertain events, and d) *Finance*: portfolio, efficient frontier.

A rule-based simulation engine supports hypothesis confirmation and provides insights into emergent behavior. A Decision Analysis approach conjugates scheduling, risk modeling, sensitivity, scenarios and finance.

PetroVR was written in Smalltalk in 1996 and rewritten in 2005-6. It comprises 5,400 classes, 80,000 methods and 480,000 lines of code (LOC). Design and coding styles conform to well-known best practice patterns, exhibiting excellent complexity metrics; e.g., 65% of its methods take no arguments, 91% one at most.

Conventional indicators poorly reflect the scope of functionality. For instance, the number of classes and LOC looks moderate. In contrast, the suite supports interoperable modeling of engineering, planning, production, risk assessment, financial and decision analysis.

Product development is highly dynamic. Three annual releases add on average 10 substantial enhancements plus 300+ improvements each. A strong refactoring culture keeps the overall size of the code fairly constant over time.

The development model is essentially agile: small team (5½ full-time smalltalkers), ubiquitous domain language, short iterations, shared code ownership, pair programming, exhaustive peer revision, constant refactoring, continuous integration.

Automatic testing is rigorously applied: 7,000 unit tests are a precondition of daily integration, covering 80% of existing methods.

Documentation policies depart from agile values in that implementation-independent abstractions are extensively discussed in 250 documents. This is often rendered necessary by the fragmentation of the available information; thus completed, the knowledge base provides developers, testers and domain experts with a coherent background of mutual understanding.

## The quality control process

The quality control process has control activities for every deliverable of the development process: specification documents, online help, release notes and executable code. The process includes 5 main activities:

1. Specification document review
2. Test case definition
3. Test case execution
4. Help & release notes review
5. Ad-hoc testing performed by domain experts

Testers perform a review of specification documents in order to become familiar with the functionality to be tested. This activity ensures the testability, completeness and consistency of the specification. During this activity testers, developers and domain experts exchange opinions about the product and the specification documents. In other domains this kind of activity is not necessary because the functionality being tested is easily learned, even by less experienced testers.

Test case definition comprises two levels: GUI testing and simulation testing. GUI testing in this context is no different from that performed in other domains. Simulation testing is a more complex task and is critical for the quality of the product.

Less experienced testers usually work on GUI testing as a way to learn about the product. Simulation testing requires more experienced testers. In order to find unexpected behaviors during this particular kind of testing, a tester has to be able to interpret statistical results and to understand specific terminology.

A divide-and-conquer strategy is used in simulation testing. Test cases are grouped by objects and their features. This strategy allows two things: (1) to narrow down the model complexity to smaller units, and (2) to have a clear idea of how much functionality is covered. After defining the cases for each root object, it is necessary to consider their combinations and discard those that are redundant (or not relevant). Here the tester's domain knowledge is critical because it is impossible to test every combination of features and objects.

The execution activity involves data preparation, model execution and result analysis. Data preparation and result analysis are the most important tasks.

Data preparation consists of defining models in a way that defined test cases are covered. These models should be complex enough to test the functionality (test cases) and simple enough to easily detect errors. Unlike tests in other domains, testers spend most of their time designing the models as these constitute a library of knowledge that will be useful in performing tests not only in this release, but also in upcoming ones.

Analyzing the results to determine whether they are correct is hard. It involves the manipulation of a large number of formulas and algorithms. A key issue here is the way testers document their findings. It is necessary to document not only the data used but also the calculation method involved. This is the only way the developer can follow the computations in order to reproduce the behavior.

Additionally, a regression test is executed in order to ensure backcompatibility. This is accomplished by running models created with previous versions of the product. The results obtained with the current release must be equal to those of previous ones.

Help & release notes review is straightforward. As in any complex and highly specific software product, technical documentation is essential for users. Writing style, completeness, formula correspondence between software and documentation are some of the validation criteria used in this type of review.

The last activity of the quality control process is the ad-hoc testing carried out by domain experts. This step is usually performed by business analysts in real-life conditions. Experts test the application with their specific problems in mind.

Experience has shown that placing this type of testing at the end of the quality control process allows detection of unintended behaviors that might otherwise go unnoticed. Note however that real-life cases tend to be overly large and complex. In order to incorporate this contribution into a systematic coverage, professional testers can reduce these cases to isolate emergent issues and transfer them to a controlled environment.

## Some conclusions

Over more than three years some interesting conclusions have been drawn:

The first concerns development. Whether completely error-free software is attainable or not, it is a fact that it cannot be achieved without traditional testing. Even if we have invested time and energy in carefully defining the development processes and creating automatic tests that reach a high level of coverage, the involvement of a testing team remains essential.

Secondly, project managers should believe in devoting time to training testing teams in domain-related subjects. They should focus not only on selecting a proficient development team but also on securing the participation of a testing team with a sound knowledge of the domain.

Third, investing in knowledge management is fundamental. The corpus of knowledge in this type of industry is not static: it inevitably undergoes constant change, and so do work teams. A good knowledge management ensures that this ongoing renewal of both information and human resources does not result in loss of knowledge for the project.

Fourth, automatic testing can eliminate recidivism. Random samples of test cases showed that 0% of already-addressed failures reappear. (2% of functional tests that originally passed fail one year later.)

Finally, in a domain so characterized by the need for analytical skills, the role that testers must assume in the process cannot be limited to early detection of failures. The active involvement of testers in exchanges with developers becomes indispensable, as their feedback may even shed light on design issues.

*We are indebted to Diego Seguí for his assistance in putting this paper together.*

## > biography

**Leandro Caniglia**
*has worked at Caesar Systems since 2001 and serves as chief technologist and director of software development. For more than a decade prior to joining Caesar Systems, Caniglia worked as a Smalltalk consultant for several companies in Argentina, Brazil and Chile. He was professor at the University of Buenos Aires for more than 20 years. Caniglia has also worked as a researcher in the CONICET, the official office for scientific research in Argentina. In 1997, he founded the user group SUGAR. He has a Ph.D. in Mathematics and has published extensively on Computational Algebraic Geometry. In 2007 Caniglia became President of FAST, the organizing board for the Annual Argentine Smalltalk Conference.*

**Damián Famiglietti**
*has worked at Pragma Consultores since 2000. He is a Quality Assurance expert. He has 10 years of solid professional experience in Software Quality Assurance. Damián has led testing teams in several projects of different industries. He holds a Systems Analyst degree from the University of Buenos Aires (Argentina).*

**Ernesto Kiszkurno**
*has worked at Pragma Consultores since 1996 and became partner in 2007. He serves as director of Research & Development and is a Quality Assurance expert. Kiszkurno has 15 years of solid professional experience in the areas of software engineering, process improvement, project management and quality assurance. His main activity in recent years has been the training, leadership and management of testing teams at multiple companies and in various industries. He holds a Bachelor of Computer Science degree from the University of Buenos Aires (Argentina) and a Master in Business Administration degree from the Torcuato Ditella University. He has also been vice president for Hispanic America of the Software Testing ISTQB Qualifications Board since 2008. For more information, please visit www.ernestokiszkurno.com.ar.*

iqnite DEUTSCHLAND2011

# The Conference for Software Quality

24 – 26 May 2011 | CCD Congress Center Dusseldorf


Conference Theme: **Industrialisation – Software Quality 2020**
The conference programme is online.

On the following pages, please find a selection of our renowned **iqnite** sponsors.

**iqnite-conferences.com/de**

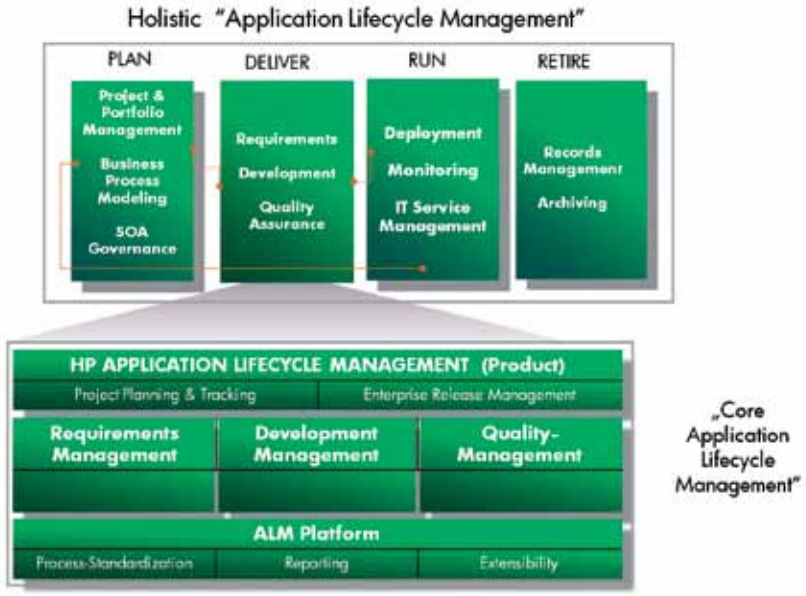
# The Core Application Lifecycle under control

**Efficient Application Lifecycle Management (ALM) requires an integrated solution that covers not only the software development but also the communication with upstream and downstream processes, for example with Project and Portfolio Management (PPM) and IT Service Management (ITSM).**

ALM is often reduced to the software development process itself (Software Development Lifecycle, SDLC): from the definition of the requirements, through development including source code and developer task management to quality assurance. HP, on the other hand, advocates taking a significantly broader view. First of all, only a closed chain from planning, through development, all the way to the operating phase permits a real orientation to business. Secondly, a focus that is too narrow tends to lead to the formation of silos – while potentials for efficiency remain dormant, especially in the close interplay of the teams. The greatest loss of time – and thus financial losses – always arise in ALM when tasks are passed between the teams involved, in particular when these teams use different solutions and are unable to access commonly available project information.

## Core Lifecycle versus Complete Lifecycle

HP makes a distinction between the core and complete application lifecycles. Even the narrow core lifecycle is not equal to the SDLC, rather is significantly wider: alongside the four main areas 'Requirements', 'Development', 'Quality Management', and 'Performance Management', HP regards the handover mechanisms to neighboring disciplines as a major part of the core application lifecycle. Here, it is crucial to link Project and Portfolio Management (PPM) to Business Process Management (BPM) to Requirements Management, and down to quality assurance not only with regard to functionality but also with regard to performance, security, guideline conformity and, ultimately, the transfer to operation.



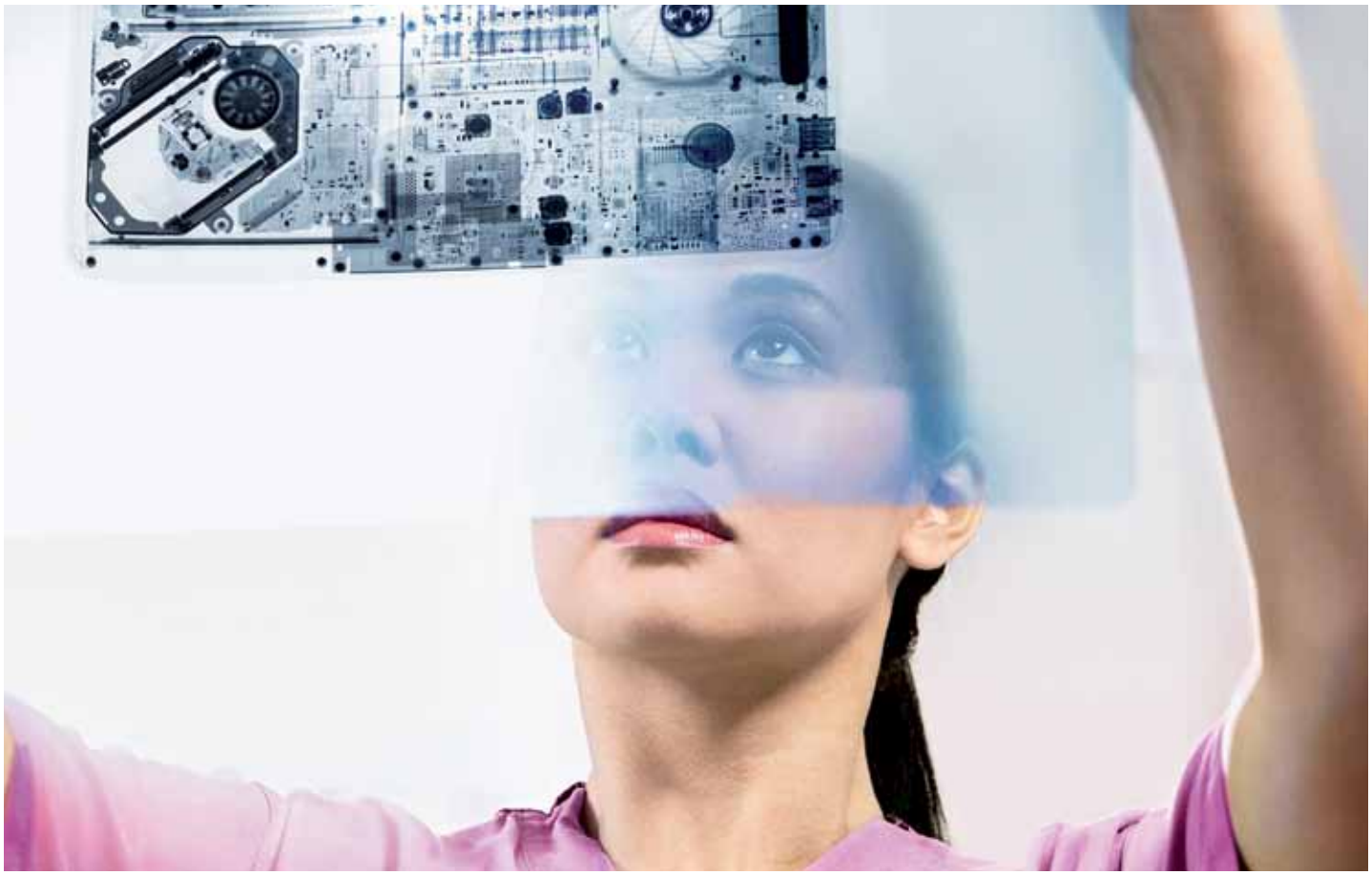

HP ALM 11 provides an integrated platform for management of the core application lifecycle with interfaces to cover the complete lifecycle of applications. The solution integrates seamlessly into the market-leading development environments and provides possibilities for the automation of test procedures as well as the required flexibility for deployment with distributed development teams.

Learn more on HP's ALM 11 solution on **www.hp.com/go/alm**

Download HP's ALM Handbook on
**www.computerwoche.de/subnet/hp-instant-on/app-modernisierung**

## Visit HP Software, Premium Sponsor, at iqnite 2011, May 24-26, Düsseldorf!

iqnite DEUTSCHLAND2011



# Quality Assurance in complex IT- and Business Process Environments.

T-Systems Testing Services minimize the operational risk in large scale IT projects and secures your IT investment.

Visit us at the T-Systems booth at the iqnite 2011 to learn more about the T-Systems Testing Services offering!

## Test & Integration Services (TIS).

T-Systems provides a complete spectrum of testing and integration services. Scope of the services are:

- Test Management and Project Management
- Functional and non-functional Tests
- Load and Performance Tests
- Test Automation
- Test Consulting
- Test Infrastructure Management

## Test & Integration Center (TIC).

A Test & Integration Center (TIC) assures software quality in complex IT and business process environments . It optimally secures the going-live and operations of new and updated IT systems, especially in large scale and complex environments. A TIC comprises at least test management and test execution services but may also be enhanced by other supporting services.

**Contact:** testing-services@t-systems.com, T-Systems International GmbH, Marketing, Böhringer Str. 50, 78315 Radolfzell, Germany

T · · Systems · · ·

# Good prospects.
# For your objectives.

SQS AG – Premium Sponsor
Visit our booth at iqnite 2011

iqnite DEUTSCHLAND2011

The SQS Group (SQS) is the largest independent provider of software testing and quality management services. SQS's success is based on almost 30 years of business experience, which has resulted in satified customers for over 5,000 projects.

**Grow with us!** SQS currently has over 150 vacancies for consultants, test managers and testers throughout Europe. Decide on SQS – for a high level of security with excellent career prospects. SQS supports and encourages every single employee.

Further information and individual job descriptions can be found online.

**www.sqs-group.com | www.sqs-aussichten.de**

Excellence through Independence

SQS SOFTWARE QUALITY SYSTEMS

iqnite DEUTSCHLAND2011

# iqnite 2011: Industrialisierung der IT

24. – 26. Mai 2011 | CCD Congress Center Düsseldorf

Einige Highlights aus dem Konferenzprogramm:

## Keynotes

Prof. Claus Lewerentz,
Brandenburgische Technische Universität Cottbus

Kann man Qualität sehen? – Softwarevisualisierungen für komplexe Kennzahlen-zusammenhänge

Bruno Kunz,
Schindler Aufzüge

Anforderungen an die Qualität von Aufzug-systemen – Mehr als nur eine Frage der Funktionalität

Dr. Georg Kraus,
Kraus und Partner

Softwareindustrie am Wendepunkt?

## Neu: iqnite Interaktiv

Pecha Kucha mit Networking

Industrialisierung aus Sicht der Anbieter im 20-Sekundentakt

SMS-Voting – Wie denken Sie?

Ein interaktives iqnite Meinungsbild

Open-Space

Open-Space – Das Programm sind Sie!

**iqnite-conferences.com/de**



# Software Testing @ Printing Software

*by Ashish Mittal*

Information technology has become an integral part of every business now. Some three decades ago, no one would have thought of a world so dependent on software. I have been associated with the printing domain for the last twelve years now. During that time this industry has utilized great IT potential, which obviously has improved efficiency, quality and ROI for print houses compared to the 20 years ago when work was done manually. I have seen exponential growth of IT in the printing industry and have been fortunate to follow the technical changes since I joined the field. Product based software companies prefer hiring domain experts into their testing teams, and so do companies making print-related software. The skill set required for a functional tester is becoming increasingly domain centric.

Page layout designing applications like QuarkXPress, photo editing and vector graphics tools like Photoshop and Corel Draw; PDF makers and editors like Acrobat Professional, raster image processors, embedded software for printers and plotters are some of the most famous examples of software applications being used in the print industry. Even though printing applications may not be as critical as health industry applications, like any software these tools have to be quality oriented, quick, user friendly and bug free to stay in the market and meet the daily needs of a printing house. Following Agile or Waterfall SDLC models, testing teams start working on products from the inception of the requirements document. Testing of such applications is certainly requirement-driven and is based on the nature of the product under development, but there are some features which are important in the print industry and henceforth are vital functional test cases when testing printing software.

A document created in any such application which is intended to be printed can have text, vector graphics, colors, images, tables and other similar elements. Unit testing of an application while using different types of these elements individually, or integration testing of all these elements together, is basic functional testing of any print related software.

**Fonts**: Font is a complete character set of a single size and style of a particular typeface. There are different types of font families available like True Type, Type 1 and Open Type fonts. Testing a particular font in printing software includes

- Typing text in an electronic document and apply the font which needs to be tested.
- The font should be displayed fine on screen in all possible font sizes.
- Applying different styles on fonts like bold, italic, color, strike-through etc. should not disturb the font edges and serifs.
- Saving, closing and reopening of a document carrying a particular font should show the last stage of font retained in the document.
- Saving the document as PDF or printing the document on a PS or non-PS printer should show the fonts perfectly fine in output.

**Pictures and graphics**: Image file formats are standardized means of organizing and storing digital information. These are composed of either pixel or vector data. Some of the most popular image formats are JPEG, TIFF, EPS, PDF, PNG, GIF and BMP. Different image formats are used for different end uses. GIF and PNG are good picks when the image is intended to be displayed on a computer screen through web browsers. JPEG can be used both for displaying on screen and for printing. TIFF, EPS, PDF are good image formats for printing purposes. Testing different image formats in printing software is an important part of the functional tests -

- To test importing or placing of different image formats in the print document.
- To test that images display well on screen, if applying attributes like blur, negative, bright, contrast, saturation etc. on different image formats.
- Saving, closing and reopening of documents carrying different images with different attributes, should show the last stage of images retained in the document.
- Saving the document as PDF or printing the document on a PS or non-PS printer should show all kinds of images perfectly fine in output.

**Output color space**: Objects, documents or images, which viewers see on computer or TV screens show in RGB color space, whereas documents, graphics or colors which are intended to be printed on paper are printed using CMYK inks. CMYK stands for cyan, magenta, yellow and key (black) colors. Most of the color hues can be produced and printed using these four inks, but there are obvi-

ously some colors like gold, silver and other metallic colors which cannot be produced by mixing CMYK inks and henceforth special colors (spot colors) are used. So when a document is printed through a professional printing application, it offers an option to the user to either print the document in CMYK only or CMYK and spot colors. Testing this feature is a little tedious but nevertheless extremely important. A bug here may cause a huge loss to press houses. A tester while testing this functionality should follow these steps:

- Create a document with images of different formats, fonts and vector objects.
  - Select those images in which RGB / CMYK / spot colors are used.
  - Apply RGB / CMYK / spot colors to text and vector graphics in the document.
- Verify printing one such document, which has different types of content in different colors, in "CMYK" output color space. Take the print out and test it using densitometer. The test will pass if all content shows colors in CMYK color space, irrespective of their color space in soft form.
- Verify printing the same document in "CMYK and Spot" output color space. Take the print out and test it using densitometer. The test will pass if the content applied with spot color retains its original color in hard copy print too, and all other colors show in CMYK color space, irrespective of their color space in soft form.

**Color management**: In digital imaging systems, color management is the controlled conversion between the color representations of various devices. Let's illustrate this with an example. You shoot a photo using a Sony digital camera and transfer it to your computer. When you open the same photo on your LG computer screen, you find that the colors are different. You ignore it, as the color difference was somewhat acceptable. You finally decide to print it on an Epson color printer and - guess what - your brown colored cap shows muddy black in print... This happened because your Sony camera captured a picture and saved color information in values of RGB, say for a particular area R= a%, G=b% and B=c%. Now when this image is transferred to an LG monitor, it read values of RGB and rendered them as per its own calibration and hence you observed a difference in shades, and when you finally printed it on an Epson printer, it rendered R= a%, G=b% and B=c% as per its own calibration. To prevent such issues, you do color management while printing a job. The basic idea of this practice is to achieve "What you see is what you get". The primary goal of color management is to obtain a good match across color devices, provided the devices are capable of delivering the needed color intensities. In the above example, opening the photograph on an LG monitor would have shown the colors correctly, if the Sony Cybershot camera profile had been applied on the photograph. Applying a profile does not change the image permanently. It is just a temporary view of the image, which shows colors being rendered as they show on the original device. Some of the very important functional test cases of CMS in a printing application are:

- Verify that applying profiles on an image results in a correct preview of the image on screen.
- Verify that printing an image with a certain profile applied on it produces correct o/p#.
- Verify that changing profiles changes the hue and shade of an image under use.
- Verify that CMS can be achieved on all formats of images.

Like any other software product, printing software too is tested using general approaches like automation testing using tools like QTP, Egg Plant, Load Runner or some in-house customized tools for functional testing, system testing, integration testing, performance testing, regression testing, alpha & beta testing. A big IT revolution has been observed in the print industry in the past decade, and I must say that testing printing software using standard testing practices has always been both fun and challenging for me.

Disclaimer: Brand names of software applications, or their logos or any specific technical or non technical term which relates to industries or domains, being discussed in this article, are proprietary of the respective brands and industry.

## > biography

**Ashish Mittal**
*is a Printing Technology graduate along with Masters in Business Administration.*

*He initiated his career as Quality Assurance personnel at Moserbaer India Ltd. Experience there made Ashish vigilant about Quality and its significance for generating business. He joined Quark in late 2004 as a Software Test Engineer and since then he is serving the Software Testing domain at different positions. He worked with biggies like Quark Media House, Global Graphics Software UK and later in India. Currently he holds as QA and Project Manager for Web Technology at Daffodil Software Ltd, India.*

*During leisure times, Ashish loves spending time with his family, writing technical articles, knowing news and doing social networking.*

# Subscribe for the printed issue!


**Please fax this form to +49 (0)30 74 76 28 99, send an e-mail to info@testingexperience.com or subscribe at www.testingexperience-shop.com:**

**Billing Adress**

Company: _____

VAT ID: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

**Delivery Address (if differs from the one above)**

Company: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

Remarks: _____

1 year subscription

☐ **32,- €**
(plus VAT & shipping)

2 years subscription

☐ **60,- €**
(plus VAT & shipping)

Date

Signature, Company Stamp

# Agile TESTING DAYS

**November 14–17, 2011**
Potsdam (near Berlin), Germany

www.agiletestingdays.com

Agile TESTING DAYS

**November 14–17, 2011**
**in Potsdam (near Berlin), Germany**

The Agile Testing Days is an annual European conference for and by international professionals involved in the agile world. This year's central theme is **"Interactive Contribution"**.

Please visit our website for the current program.


**Agile Testing Days 2011 –**
**A Díaz & Hilterscheid Conference**

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

info@agiletestingdays.com
www.agiletestingdays.com


## Tutorials – November 14, 2011

- Lisa Crispin: **"Hooray! We're Agile Testers! What's Next? Advanced Topics in Agile Testing"**
- Janet Gregory: **"Transitioning to Agile Testing"**
- Johanna Rothman: **"Making Geographically Distributed Projects Work"**
- Esther Derby: **"Dealing with Differences: From Conflict to Complementary Action"**
- Linda Rising: **"Influence Strategies for Practitioners"/ "Patterns for Improved Customer Interaction"**
- Michael Bolton: **"Critical Thinking Skills for Testers"**
- Gojko Adzic: **"Winning big with Specification by Example: Lessons learned from 50 successful projects"**
- Elizabeth Keogh: **"Introduction to BDD"**
- Lasse Koskela: **"Acceptance Testing: From Brains to Paper and Paper to Computer"**
- Jurgen Appelo: **"Agile Management: Leading Software Professionals"**

## Conference (Day 1) – November 15, 2011

| Time | Track 1 | Track 2 | Track 3 | Track 4 | Vendor Track |
|---|---|---|---|---|---|
| 08:00 | Registration | | | | |
| 09:10 | Opening | | | | |
| 09:30 | **Keynote** – Johanna Rothman | | | | |
| 10:30 | **"What Testers and Developers Can Learn From Each Other"** David Evans | **"Specification by Example using GUI tests – how could that work?"** Geoff Bache & Emily Bache | **"Agile Performance Testing"** Alexander Podelko | **"SQL PL Mock – A Mock Framework for SQL PL"** Keith McDonald & Scott Walkty | **"The roles of an agile Tester"** Sergej Lassahn (T-Systems Multimedia Solutions GmbH) |
| 11:30 | Break | | | | |
| 11:50 | **"Do agile teams have wider awareness fields?"** Rob Lambert | **"Experiences with Semi-scripted Exploratory Testing"** Simon Morley | **"Design For Testablity is a Fraud"** Lior Friedman | **"Using agile tools for system-level regression testing in agile projects"** Silvio Glöckner | Talk 5.2 |
| 12:50 | Lunch | | | | |
| 14:20 | **Keynote: "Who do You Trust? Beware of Your Brain"** – Linda Rising | | | | |
| 15:20 | **"Top Testing Challenges We Face Today" "** Lloyd Roden | **"Session Based Testing to Meet Agile Deadlines"** Mason Womack | **"Unit testing asynchronous JavaScript code"** Damjan Vujnovic | **"Automated Functional Testing with Jubula: Introduction, Questions and Answers"** Alexandra Imrie | Talk 5.3 |
| 16:20 | Break | | | | |
| 16:40 | **"I don't want to be called QA any more! – Agile Quality Assistance"** Markus Gaertner | **"TDD with Mock Objects: Design Principles and Emerging Properties"** Luca Minudel | **"Agile on huge banking mainframe legacy systems. Is it possible?"** Christian Bendix & Kjær Hanse | **"Automated testing of complex service oriented architectures"** Alexander Grosse | Talk 5.4 |
| 17:40 | **Keynote** – Lisa Crispin & Janet Gregory | | | | |
| 18:40 | Closing Session | | | | |

## Conference (Day 2) – November 16, 2011

| Time | Track 1 | Track 2 | Track 3 | Track 4 | Vendor Track |
|---|---|---|---|---|---|
| 08:00 | Registration | | | | |
| 09:25 | Opening | | | | |
| 09:30 | **Keynote: "People and Patterns"** – Esther Derby | | | | |
| 10:30 | **"About testers and garbage men"** Stefaan Luckermans | **"ATDD and SCRUM Integration from a traditional Project methodology"** Raquel Jimenez-Garrido | **"Test automation beyond GUI testing"** H. Schwier & P. Jacobs | **"Micro-Benchmark Framework: An advanced solution for Continuous Performance Testing"** Sven Breyvogel & Eric Windisch | Talk 5.5 |
| 11:30 | Break | | | | |
| 11:50 | **"Do we just Manage or do we Lead?"** Stevan Zivanovic | **"Agile ATDD Dojo"** Aki Salmi | **"Make your automated regression tests scalable, maintainable, and fun by using the right abstractions"** Alexander Tarnowski | **"Beyond Page Objects – Building a robust framework to automate testing of a multi-client, multi-lingual web site"** Mike Scott | Talk 5.6 |
| 12:50 | Lunch | | | | |
| 14:20 | **Keynote: "Haiku, Hypnosis and Discovery: How the mind makes models"** – Liz Keogh | | | | |
| 15:20 | **"A Balanced Test Strategy Strengthens the Team"** Anko Tijman | **"Effective Agile Test Management"** Fran O'Hara | **"Sustainable quality insurance: how automated integration tests have saved our quality insurance team."** Gabriel Le Van | **"Automate Testing Web of Services"** Thomas Sundberg | Talk 5.7 |
| 16:20 | Break | | | | |
| 16:40 | **"Testing your Organization"** Andreas Schliep | **"Get your agile test process in control!"** Cecile Davis | **"Measuring Technical Debt Using Load Testing in an Agile Environment"** Peter Varhol | **"Real loadtesting: WebDriver + Grinder"** Vegard Hartmann & Øyvind Kvangardsnes | Talk 5.8 |
| 17:40 | **Keynote: "Five key challenges for agile testers tomorrow"** – Gojko Adzic | | | | |
| 19:00 | | | | | |

## Collaboration Day – November 17, 2011

| Time | Track 1 | Track 2 | Track 3 | Track 4 |
|---|---|---|---|---|
| 08:00 | Registration | | | |
| 09:25 | Opening | | | |
| 09:30 | **Keynote: "No More Fooling Around: Skills and Dynamics of Exploratory Testing"** – Michael Bolton | | | |
| 10:30 | **Open Space** – Brett L. Schuchert | **Testing Dojos** – Markus Gaertner | **Coding Dojos** – Markus Gaertner | **TestLab** – B. Knaack & J. Lyndsay |
| 11:30 | Break | | | |
| 11:50 | **Open Space** – Brett L. Schuchert | **Testing Dojos** – Markus Gaertner | **Coding Dojos** – Markus Gaertner | **TestLab** – B. Knaack & J. Lyndsay |
| 12:50 | Lunch | | | |
| 13:50 | **Keynote: "Stepping Outside"** – Lasse Koskela | | | |
| 14:50 | **Open Space** – Brett L. Schuchert | **Testing Dojos** – Markus Gaertner | **Coding Dojos** – Markus Gaertner | **TestLab** – B. Knaack & J. Lyndsay |
| 16:50 | **Keynote: "The 7 Duties of Great Software Professionals"** – Jurgen Appelo | | | |
| 17:50 | Closing Session | | | |

# Business demand versus regulatory change: understanding the conflicts

**A discussion on how investment banks manage the testing of regulatory change whilst keeping pace with market conditions and leading edge technologies**

*by Paul Boston & Roger Fox*

Following the 2008 liquidity and credit crisis, investment banking regulators across the world have continued to introduce unprecedented restrictions on existing capital adequacy requirements and associated liquidity risk management practices.

Different regulators and regulations apply to operations in different parts of the world. Most major investment banks have international operations operating in accord with 24x7 markets. Changes to the regulatory framework are in conflict with the unprecedented speed at which financial institutions are seeking to maximize their profits, cut the cost of their business operations and embrace the opportunities presented by changing technology.

From the board level and trading floors, through to the compliance and audit departments, the implications of these new regulations must be addressed. The fast-paced and demanding trading room environment requires quick fixes to be made to circumvent the limitations of the existing technology, whilst ensuring minimal client impact. In contrast, down-stream business functions require more structured and thorough approaches to ensure that the banks do not become exposed to malpractice and unwanted attention from the authorities.

This article considers the internal conflicts and challenges that investment banks are facing, and how a strategic approach to testing the required software changes can alleviate some of the problems being faced.

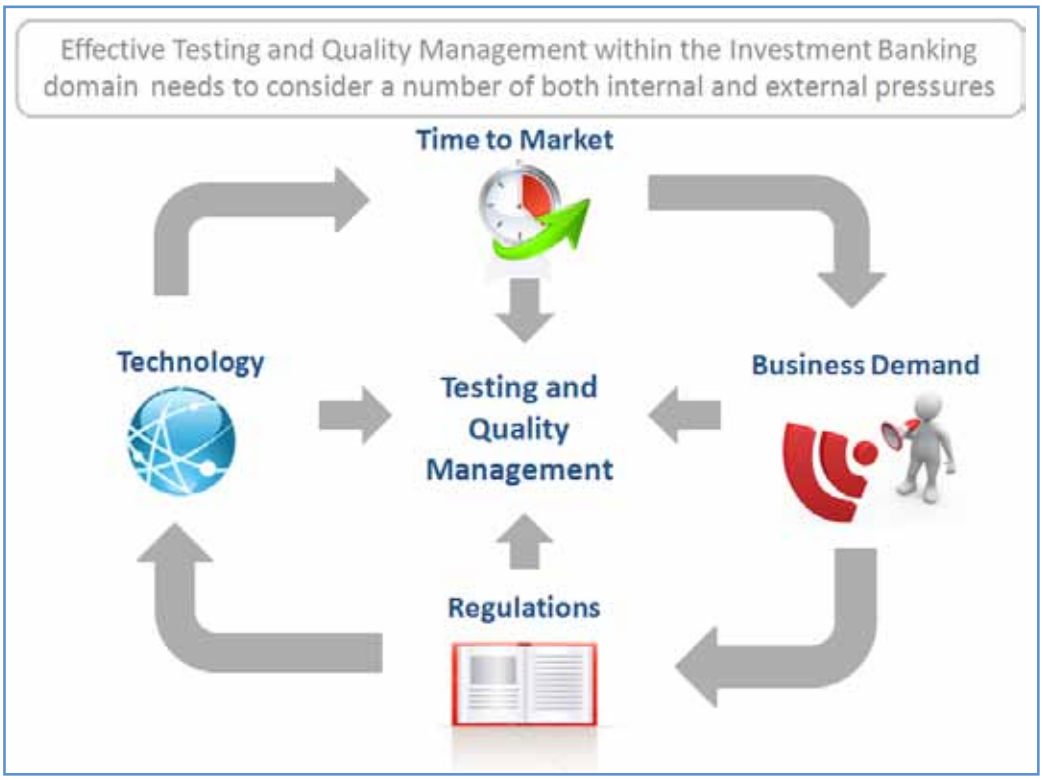



Figure 1: Considerations for effective testing

It should be noted that all investment banks are different in attitudes, culture, technologies and programme management techniques. The recommendations discussed here therefore need to be viewed alongside these other considerations.

## Interpreting regulations and mandates

Certain Basel III requirements, such as the **Expected Positive Exposure** mandate will dictate software or system changes. The implications of the software changes upon applications will need to be investigated and applied across all the impacted systems. The nature of the new regulatory requirements is such that they require a complete understanding of both the technology infrastructure and internal trade processing capabilities to ensure that the changes are fully understood. As a result, the key to successful implementations of this nature is to deploy project and testing teams that can demonstrate domain knowledge that complements the internal system capabilities of the bank.

One key issue faced by investment bankers is the apparent conflict between regulations, as yet unresolved by case law. For example, the Data Protection Act 1998, the Freedom of Information Act 2000, and the Data Retention (EC Directive) Regulations 2009, each have different requirements on data retention. Similarly Sarbanes-Oxley imposes different retention rules for data used in the US.

System architects and technology teams may not be aware of such regulatory requirements; their focus is more likely to be on testing and optimizing the system performance of real-time trading systems. Latency of an application at one institution may cause an open trade offer to be taken by a competitor with a better designed and faster system. However, testers of such applications will also need to take decisions on compliance, and make provisional and conditional assessments of fitness for purpose, often pending the advice from internal compliance and audit departments who may be unable to provide such advice in the required timescale.

## Achieving testing objectives in a rapid delivery development

It is an accepted fact that the majority of systemic changes within today's market place are both complex and challenging, and that meeting the demands of the various business areas places unwelcome pressure on both project and testing teams. The nature of the trading room environment tends to attract high profile individuals that are demanding of themselves, their colleagues and the supporting areas of the financial organization. Of all the challenges that are experienced today, one of the most common is related in some form to the variety of the **different personalities and associated characteristics** of the people that are employed by the banks.

The majority of change programmes originate from one or maybe two business departments rather than an entire financial organization. The trend is that the originating department is likely to benefit from the change itself, and therefore has a direct interest in pressing for the **Time to Market** (implementation timescales) **to be as short as** possible, to ensure that the cost of change is kept to a minimum. Other departments tend to have an indirect interest in the change programme, but are likely to view additional project work as a burden and unnecessary overhead in terms of both resource utilization and workload.

Whilst traders and front office staff expect rapid delivery, the reality is that the development and testing of projects requires detailed planning and scheduling of resources. In achieving this, it is necessary to engage business managers with sufficient knowledge to understand the importance of testing, and with sufficient authority to ensure that it is correctly undertaken.

Such engagement means that testing should be at the forefront of change programmes to ensure that the requirements are clear and concise, that impacted business areas are identified and engaged at the project inception, and ultimately to ensure that testing is part of the complete development lifecycle.

As most investment banking programme managers favor an agile methodology to software development, this should mean that continuous testing is performed throughout. Experience, however, shows that true test driven development is not fully understood, and that testing often takes place towards the end of a project, when deadlines are fast approaching and timescales are cut short. As a result, insufficient time is made available for a comprehensive phase of testing, and as testers find defects in the software, subsequent delays are attributed to the testing teams rather than the overall project management.

## Front office fluidity versus regulatory constraints

Traders and sales personnel do not typically view testing as part of their day-to-day responsibilities and will often require coaching or at least guidance on testing best practice. Operations personnel tend to take a far more pragmatic approach that embraces the need for a comprehensive testing plan. These different views can result in a conflict of effort and timescales that result from the minimal to the extended testing coverage throughout the lifecycle of a project.

Whilst the majority of business areas that support the trading systems prefer a risk mitigation policy, the requirement for rapid system change to support business demand, can result in conflicts that potentially have damaging implications to both the project and the testing teams. An example scenario is where a new trade type is implemented as an exceptional item, bypassing most internal audit procedures and without proper notification to ancillary back office accounting and transparency requirements. It will then be integrated again, as formal due diligence processes sanction the changes.

In this scenario, the solution is to ***deploy a strong test manager with both technical knowledge and a good understanding of testing best practice. However, the most important quality in these circumstances is for the test manager to possess strong stakeholder management techniques***. It is always important to engage all stakeholders at project inception, rather than just those that are directly impacted by the system or software change.

## Technology constraints and considerations

In recent years, the demand for technological innovation has increased, particularly with the advent of rich internet applications (RIAs) that dramatically improve usability and access to bespoke user-data. As a result, technology change is no longer the preserve of the technology experts, with increasingly technically astute end-users now demanding more from the applications which populate their order books and provide them with market data.

The user experience encountered elsewhere, such as on smartphones, social networking sites and associated applications are

now setting the bar for user interaction. The testing implications of this are immense, with new RIAs often providing multi-layered complexity that requires thorough testing, such as trading websites that employ Adobe Flex technology to provide complex solutions for both trading and information distribution.

The implications of utilizing these emerging technologies means that testing needs to remain one step ahead and be aligned with rapid development practices, in order to accommodate consumer driven technologies. As a result, the following testing challenges need to be considered:

- Not to be distracted by the 'rich internet' of a RIA; the front-end is important, but so is the way in which these new applications hook into legacy systems
- Ensuring the new approaches are technically robust enough to cope with:
  - Security challenges
  - Changes to the back-end
  - Architecture changes
  - Requirements to test hand-held devices (and the associated security implications)

The development of tools to automate the functional testing of new technology lags behind the speed of change of the technology itself. This is attributable to the conflict between the new technology taking market share, and the longer term development of tools that evolve to thoroughly test its functionality#, especially in an enterprise environment. This conflict is evidenced within investment banking, whereby the development and use of automated testing tools for Flex from Adobe falls far behind the take-up and use of Flex itself. Whilst potentially opening new opportunities for the developers of automated testing tools for RIAs based on Flex, their adoption within investment banks is slow.

Much has been written about the migration to 'cloud computing' and its suitability for testing. Such technology is rarely used within investment banking, particularly as a result of strict regulation regarding the confidentiality of banking information. Whilst some banks are adopting a 'private cloud' approach to some information and services, this emerging technology is still in its infancy. Test analysts must be aware of the sensitivities of bankers to the risk of data loss, and it is a credit to investment bankers that sensitive information appears more likely to be lost by public services than by banks. However, in circumstances where restrictions are not imposed, individual traders are able to use cloud services without sanction; thereby risking the breach of data protection and other regulations.

Testing across multiple platforms and technologies does not lend itself to a traditional end-to-end test strategy, as quite often **new technologies tend to conflict with legacy systems**. For example, where new 'publish and subscribe message technologies' are employed using specific 'peer-to-peer authentication', it may be very difficult to identify which services consume internally broadcast messages or transactions. The considered solution in this situation is to assume a three-phased approach:

### *1. Integrate*

Systems integration testing is a complex task in many investment banks, where transactions and data are shared across numerous disparate systems. The initial phase of testing is to ensure that information flows correctly in each direction of the test environment. Normal methods would include an initial 'smoke test', which can be executed without directly affecting any of the data stores. These tests are often automated, with a series of parameters allowing the smoke tests to be conducted from the same test framework as the later functional testing. Such smoke tests can be employed as a quality check to ensure that the correct instances of systems and applications are communicating with each other.

The goal is to be assured that neighboring systems are actually able to communicate correctly. Whilst being no different to standard integration testing in other domains, there are very particular demands made in investment banking to ensure that ALL affected systems form part of the tests, and that test execution always produces a permanent trail to show where data is modified as part of the testing.

### *2. Differentiate*

The test approach should be built or chosen for its ability to demonstrate differences in method. Test automation tools employed in this approach must differentiate between the key layers and technologies. Some will address transactions at a message level, testing key functionality at the transport layer. Others will address data storage, speed of access and the ability to trace specific transactions from capture to their effect on the bank's balance sheet, profit and loss accounts, risk, plus applications alerting those responsible for disclosure to the appearance of unusual customer transactions. From the outset, the tools and testing methods must be designed to accommodate the various needs of different (and sometimes opposing) parts of the business operation, providing complete coverage. In addition, the tools must differentiate between the specific features of the different technologies that have been used.

### *3. Mitigate*

In order to mitigate associated risk, there are numerous avenues for risk-based testing. When considering the investment banking domain, we tend to consider the implications of defects from a number of risk mitigation aspects including; financial risk, operational risk and reputational risk. With the very large financial value often involved in trades, risk mitigation can take a number of forms in business operations, where speed of response is a key differentiator of margin. Regulations have slowly adapted to require more and more liquidity to cover open positions, measured by the risk of counterparty failure. These are all factors to be mitigated in business, which must be mirrored in testing.

## Changes to business processes and procedures

In addition to software and system changes, the introduction of Basel III will demand a number of enhancements to business processes and procedures, including most notably the mechanics that the banks will need to adopt in order to conform to the changes in liquidity ratio regulations. This will entail the monitoring of the bank's exposures to different counterparties and clients.

The consequences of this type of change are more in line with how investment banks embrace change. Whereas system modifications ultimately enforce a conscious change, amendments to procedure or regulation require indirect change that the bank will have to monitor internally.

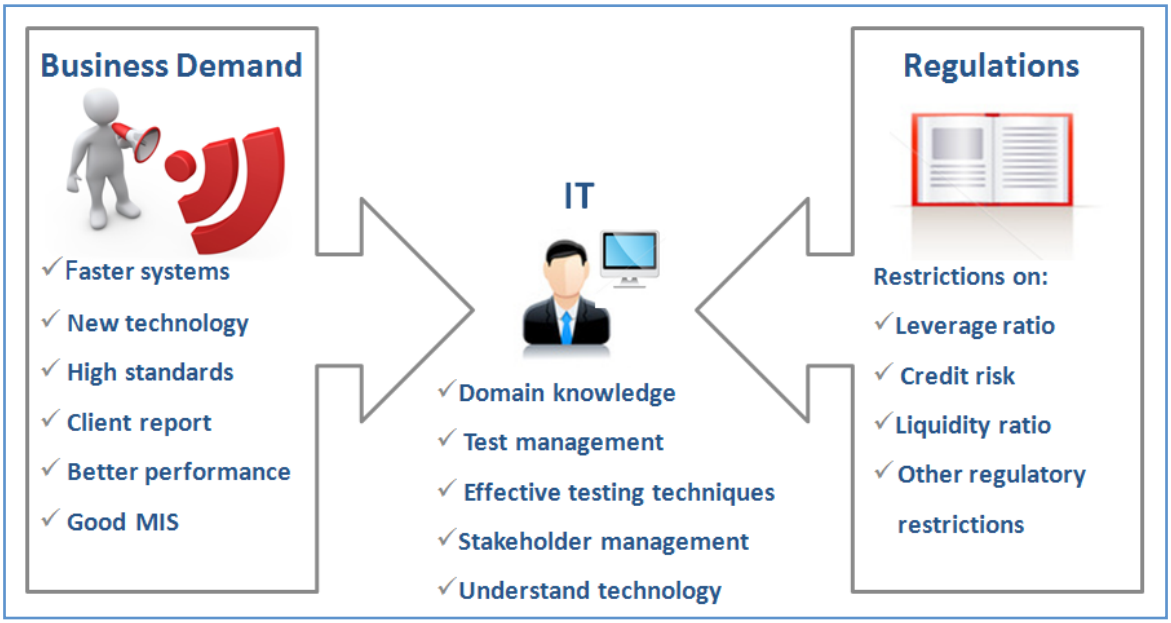



Figure 2: The pressures on the Test Manager

## Conclusion

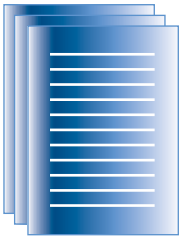
A natural conclusion to the majority of the challenges discussed would be to adopt a heavy-handed management approach to ensure that mandatory changes are enforced within investment banks. Whilst a number of financial organizations are likely to take this approach, this does not tend to embrace best practice, or the requirement to integrate testing efforts across the organization. The recent experience of City Index in being fined £490,000 (GBP) for failing to keep adequate records of transactions demonstrates that, whilst their technology was fit for purpose at a trading level, the associated back-end data retention fell short of required standards. This was a clear failure of both IT and the business in conforming to explicit regulations.

***Testing should be viewed as the critical measure of quality and project success in any IT project. This is more likely to be successfully achieved by ensuring that from the outset the testing function is seen as the pivotal project component that brings IT and the business together.***

### > biography

**Paul Boston**
*Paul is employed as a Principal Consultant within the Rule Financial testing practice. With over 25 years experience of working in the City, Paul has previously worked at a number of Tier 1 investment banks, software companies and consultancy firms that support the Investment banking community in the UK and across Europe.*

**Roger Fox**
*Roger is employed as a Test Analyst within the Rule Financial testing practice. He has over 25 years experience in IT, beginning in a development team and progressing through a number of investment banks and consultancies. His breadth and depth of experience has led to engagements within the front, middle and back offices of many Tier 1 investment banks*



# Distributed Testing Models

*by Rahul Verma*

In the first article – Test Encapsulation - published in the Q3/2010 edition of Testing Experience, I explained how the design of a test automation framework (from now on referred to as TAF) should start by thinking of tests first and then the surrounding features. In this article I'm going to address another challenging area in the design of TAFs – **Distributed Testing**.

Distributed testing is about enabling the TAF to distribute and run the same or different tests across multiple test machines at the same or different time(s), monitor their status and consolidate test results from all test machines.

In distributed testing, the test framework is logically split into two types of machine:

- The Test Controller (from now on referred to as **Controller**) – The machine which distributes the tests, monitors their overall status and is responsible for results consolidation.
- The Test Runner Machines (from now on referred to as **Runners**) – The machines which run the tests.

In certain situations, a machine could very well be the Controller as well as a Runner. Accordingly, it can have logic corresponding to either one or both functional areas.

Some typical uses of distributed testing are:

- **OS Platform Coverage** – Running tests on various platforms
- **Hardware Coverage** – Running tests on machines with different hardware or resources
- **Performance Testing** – Generating simultaneous requests to a target application because a single machine would not be able to generate the target load
- **Security Testing** – Separating the Runner from the Controller for the basic reason that such tests can result in operating system crashes
- **Reducing Test Cycle Time** – Even though the tests can be run on a single machine, multiple similar machines are used to allow parallel execution
- **Common Test Framework** – Running various types of test on different machines because they need different configurations.

Through the course of the next sections, we will try to understand the challenges faced in the distributed testing mechanism and various ways of developing an automated solution for the same.

## Some Design Questions in Test Distribution Mechanism

Having understood the basics of distributed testing, let us have a look at various questions that we need to answer while designing a TAF that supports distributed testing.

- **Test Registration** – Can tests be registered with the Controller so that they can be logically represented for distribution purposes?
- **Test Allocation** – Can the TAF allocate a subset of tests (a subset could be as big as the superset) to a Runner?
- **Test Scheduling** – For a given test cycle, can the Controller support building a test schedule for Runners? Does it support multiple products/builds under test at the same time? Does it support assigning priorities to some tests/ builds/ products/ requesters? Does it support a priority override in case an otherwise low priority run is to be given a higher priority?
- **Test Configuration** – In addition to tests, can the Controller support configuration options e.g. build under test, verbosity of reporting etc.?
- **Test Trigger** – How is a test cycle triggered? Is it with the push of button/command with manual intervention or an auto-event? Who triggers a test on the Runner – the Controller or the Runner itself?
- **Launching the Runner** – If a Runner was shut down after the previous cycle, can the Controller bring it up when required?
- **Test Schedule/Configuration Sharing with Runner** – Test scheduling is done on the Controller. How would a Runner know what has been scheduled for it?
- **Communication Protocol** – The Runner has what the Controller sent to it. Can the Runner interpret the instructions?
- **Ensuring Runner has the Required Resources** – The Runner knows what tests it must execute. How do we make sure that it has got all the resources – latest version (or a given version) of test code, build under test, test data etc.
- **Monitoring Runner Status** – How does the Controller know at what stage of test execution a Runner is? How does it know if the test execution has been halted due to an issue e.g. in case of a crash?
- **Test Results Sharing With Controller** – How and when does a Runner send the test results back to the Controller? Does the Runner have different stages of results sharing varying in contents?
- **Results Consolidation** – Can the Controller consolidate results from all Runner machines and show the overall status during and post execution?
- **Continuous Execution** – Once a Runner has finished executing a given allocation of tests, can it pick up the next set on its own from the Controller, or can it be instructed by the Controller to do so without manual intervention?
- **Aborting Test Execution** – Can the Controller abort test execution mid-way on a Runner?

It's pretty obvious that the above questions cannot be dealt with within a single article. I will write on all these areas in detail in future articles in this series.

In this article, we will look at the fundamentals of distributed tes-

AgileReq

## September 9–11, 2011 in London, UK

## The Conference for Product and Requirements Management in Agile

### Call for Papers

The call for papers ends by April 15, 2011.
The theme of this year's conference is **"Building software that matters"**.

Go to **www.agilereq.com** and submit your paper!

Topics of interest:
Emerging practices for managing requirements and specifications in Agile/Lean processes · Release planning in Lean and Agile · Evolutionary product management · Agile business analysis · Providing big-picture visibility and feedback to sponsors · Engaging business users and stakeholders · The roles of business analysts and product owners in lean/agile team · Product management/requirements challenges for distributed teams · Experience reports of applying user stories · Transitioning from traditional product management practices to agile ideas · Value-streams · Applying agile in regulated environments · Requirements traceability · Estimation and planning in Agile and Lean · Applying Minimum Marketable Features · Focusing development and delivery on things that really matter · Core domain distillation

www.agilereq.com

ting in terms of how we can enable the communication between Controller and Runners.

## Models of Distributed Testing

There are various ways in which we can design a solution for distributed testing. I like to think and talk about them as models, whereby a model helps us understand a rather complex subject with a high level pattern. I've implemented/designed automation solutions which follow the following models:

- Push
- Pull
- Peer
- Hybrid

To understand the models in a simple way, I will use a common analogy across all of them. Consider that there is a class consisting of a teacher and students who have to finish assignments as allocated by the teacher. Based on the nature of the teacher, mix of students, type of subject etc. the way the teacher interacts with the students or the way the students interact with the teacher and amongst themselves would be different. The teacher is analogous to the Controller and the students are analogous to the Runners in a TAF's implementation of distributed testing models.

***Note**: The following sections have references to basics of networking like daemons, ports etc. In case you are not aware of the same, I suggest you to have a quick look at*

- http://en.wikipedia.org/wiki/Daemon_(computer_software)
- http://en.wikipedia.org/wiki/Network_socket

## Push Model

***Analogy**: The class consists of students who need to be "pushed" to do their assignments. For every assignment, the teacher tells the students – "Do this assignment!", they go ahead and do their work and submit the corresponding assignments back to the teacher.*

A TAF following a Push model would do the same. The Runners wait to hear from the Controller, they never check on their own with the Controller about what they should be doing. It's the responsibility of the Controller to inform them when they have to run the tests, what tests are to be run and other configuration options.

Typically this is how it gets implemented:

- Runner "listens" on a given network port. This is achieved via a network daemon.
- The IP address/name of the Runner and the corresponding port are known to the Controller.
- Controller sends the test configuration/schedule to the Runner in response to a Test Trigger as per an agreed communication protocol
- Runner interprets the instructions and executes the scheduled tests with the configuration options passed.
- Runner sends intermittent status/final results back to the Controller as per the communication protocol. For this, there is a network daemon running on the Controller at a port known to the Runner. As multiple Runners contact the Controller, this daemon is multi-threaded.

Most of the open source and commercial performance testing tools are based on the Push Model.

### *Advantages/Disadvantages of the Push Model*

- The most critical part of this model is that the IPs/unique Runner names should be known to the Controller to push the tests.
- The IPs of the Runners should be static. To overcome this, a Push should ideally be implemented using unique names for Runner machines.
- The Runners should be up and running when the Test Trigger executes.
- Launching of machines, knowing whether they are up becomes easier because of knowledge of IPs/names.
- For a Runner to "become" another Runner from the perspective of the Controller, it would have to be renamed/IP has to be reallocated and the original Runner has to be shut down. This can be problematic in some situations when one is debugging test execution issues.

## Pull Model

***Analogy**: The class consists of students who are very much interested in doing their assignments. They do not wait for the teacher to tell them what they should do next. Every student in the class would reach out to the teacher and ask – "What is my assignment?" The teacher tells the student about it, the student works on that and submits the outcome to the teacher.*

A TAF following a Pull model follows similar dynamics. The Runners ask the Controller what tests they should run and how, the Controller sends the instructions back, they run tests as per the instructions and send the results back to the Controller.

Typically this is how it gets implemented:

- Controller listens on a given network port known to Runners. It is achieved via a multi-threaded network daemon.
- Runner sends request to the Controller. The request would include an identifier of a given Runner so that the Controller can run queries on the test schedule.
- Controller sends the instructions back as per the communication protocol.
- Runner sends intermittent status/final results back to the Controller as per the communication protocol.

This model is the one which the World Wide Web (WWW) employs. Google Search does not reach out to your browser to instruct it to search. You reach out to Google Search via your browser to do a search when you are interested.

One of the recent test automation frameworks that I have developed uses the Pull Model and proved to be very successful.

### *Advantages/Disadvantages of the Pull Model*

- Pull Model does not require the Controller to know IPs/names of the Runner machines.
- Any Runner can "become" another by using its identifier.
- No network daemons are needed on Runners.
- Controller does not trigger tests. The Test Trigger executes when a runner requests the Controller for tests.
- Launching of Runners cannot be done by the Controller because the IPs/names of machines are not known.
- To avoid continuous queries to Controller from Runners when no new tests have been scheduled, a sleep time must

be configured between successive queries by a Runner. This means that if a Runner gets a response from Controller mentioning that there are no tests scheduled for it at that time, Runner waits for a defined period of time before checking again. So effectively, there would be a delay between scheduling of tests and picking up of a schedule by a Runner, depending on the sleep time configured as well as when during that sleep time the tests are scheduled.

## Peer Model

Analogy: The class does not have a teacher. A random student is picked up and made the "instructor". Based on how the students are motivated to do their assignments, either this instructor reaches out to the rest of the students to tell them what to do, or the students reach out to the instructor to know their assignments.

A TAF following a Peer model would also behave in the same manner. The model is very similar to Push or Pull Models based on the choice made. The difference is with respect to who the Controller is. Unlike pure Push and Pull models discussed earlier, none of the test machines is labeled as the only Controller and thereby making it special from the rest of the test machines.

Typically this is how this gets implemented:

- Each Runner knows the IPs/names of all other Runners in the deployment as well as the port at which the network daemon is running on these machines.
- Runners maintain a record of the current Controller.
- When a new Controller is configured, it declares itself as the Controller to the currently active machines. This would overwrite the previous Controller record.
- When a new Runner becomes active, it can query any of the active Runners/Controller to know who the Controller is.
- If the TAF follows the Push Model, the Controller sends the instructions to its peers (Runners).
- If the TAF follows the Pull Model, all peers (Runners) would request instructions from the Controller.

### *Advantages/Disadvantages of the Peer Model*

- This model is very useful to build a robust TAF because it does not depend on a single Controller machine to configure and schedule tests. The facility is available on all Runners and anyone can become a Controller if you wish. It's a good alternative to building a back-up Controller machine.
- It's very useful when geographically split-up teams decide to use a common framework. In such situations some functionality of the Controller, for example, results archiving, builds repository, report generation interfacing, should be off-loaded to a server machine accessible to all.
- Each Runner should know the IPs/names of all other Runner machines.
- As any Runner can become a Controller, it means all machines need to be of good configuration. For example, all machines should be able to run a multi-threaded network daemon as needed by a Controller, which would need optimum hardware resources.
- A peer based mechanism should be established to know the current Controller so that when a Runner machine starts, it knows the Controller. Various alternatives exist to overcome this, but would result in a semi-Peer model (which may not be bad based on your context).

## Hybrid Model

***Analogy**: In the previous sections, we assumed that all students are alike, which is not the usual case. This class is the one which we would normally come across with all the complexities of human nature in play. Some students need to be pushed, some are enthusiastic, some pushed ones or enthusiastic ones form their own study groups.*

This is the most common situation which is faced by the developers of a TAF. Some categories of test should be triggered automatically, others need more flexibility and still others need redundancy of Controllers.

As you might have observed, many of the features discussed in the previous models are common across all of them. Differences lie with respect to where and how those features are exercised. With careful thought and a little development overhead, the model of distribution itself can be made configurable and built into the way Controllers/Runners are configured.

The Hybrid Model should not be confused with "No Model". Hybrid is not the same as chaos. Ironically, developing a general-purpose hybrid model which can be configured to enable distributed testing to support Push, Pull and Peer models, is the most challenging of all models discussed so far.

### *Advantages/Disadvantages of the Hybrid Model*

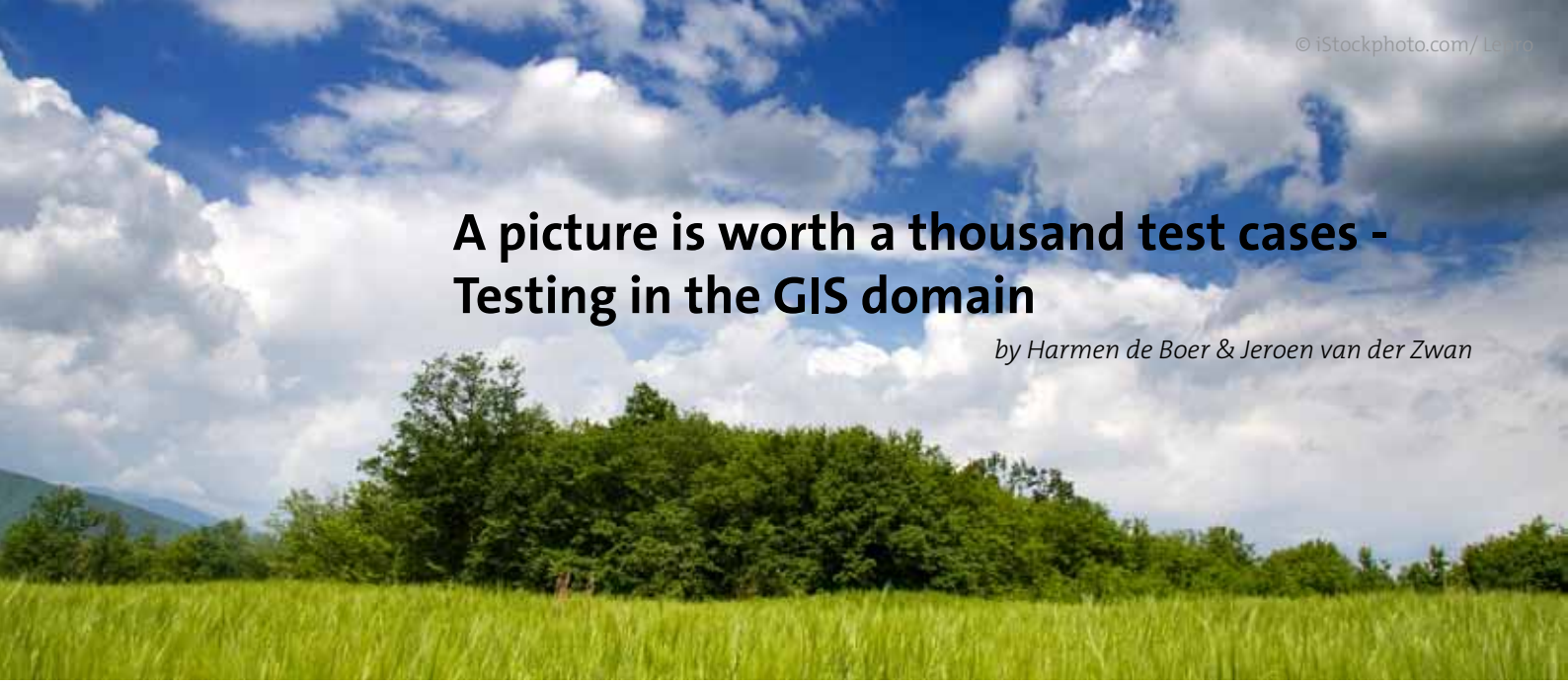
- A Hybrid model would provide you with the most flexible solution.
- It needs a lot of careful upfront design effort. You might be building something for a future situation, which may never come.
- This is the most likely candidate for being adopted by multiple teams as they can deploy it as per their needs.

## What's Next?

Next, I will write on a subset of other crucial aspects of distributed testing which were discussed at the beginning of the article.

### > biography

**Rahul Verma**
*is a Senior QA Technical Lead with the Anti-Malware Core team of McAfee Labs, India. He runs the Testing Perspective (www.testingperspective.com) website for which he was awarded the Testing Thought Leadership Award by PureConferences. He has special interest in performance testing, security testing, Python and design of test automation frameworks. Rahul has presented at several conferences, organizations and academic institutions including GTAC, CONQUEST, STeP-IN, ISQT, TEST2008, Yahoo! India, McAfee and IIT Madras. He is a member of the Working Party for the ISTQB's Advanced Level Test Certifications.*
*rahul_verma@testingperspective.com*




# A picture is worth a thousand test cases - Testing in the GIS domain

*by Harmen de Boer & Jeroen van der Zwan*

In everyday life we are surrounded by Geographic Information Systems (GIS). When we look at a weather forecast or use a car navigation system, we are indirectly or directly using a GIS. Complex GIS applications are used throughout a wide range of businesses, from defense and disaster management to the transportation and utility sectors.

What makes GIS different from other information systems is the spatial component. All spatial data is related to one or more coordinates, they have specific locations in space and time. The spatial data is used to give a geographical representation or picture of the subject which is analyzed. Current structural testing methods fall short when testing a GIS in comparison with regular applications. The domain of GIS needs a different approach from the testing perspective.

This article will explain what a GIS is, what problems are encountered when testing a GIS, and what initiative is taken to tackle these problems in order to professionally test a GIS with adjusted testing methods. These general GIS testing methods are currently developed at Alliander and Bartosz based on lessons learned from practical experiences. One technique will be discussed more in depth as an example.

## What is a GIS?

To understand more about the domain of GIS and testing, one needs to know what a GIS is. Hendriks and Ottens [1] define GIS as follows: A geographical information system is "...a computer system which offers tools to structure, save, edit, manage, question, analyze, and visualize related spatial and non-spatial data in such a way that the data will offer valuable information to answer a certain policy or research question". Depending on the context in which a GIS is used, certain aspects of this definition can play a more important role than other aspects. At the core of a GIS is the relation between spatial and non-spatial data, it is this relation which separates GIS from other administrative systems and lies at the heart of the GIS and testing domain.

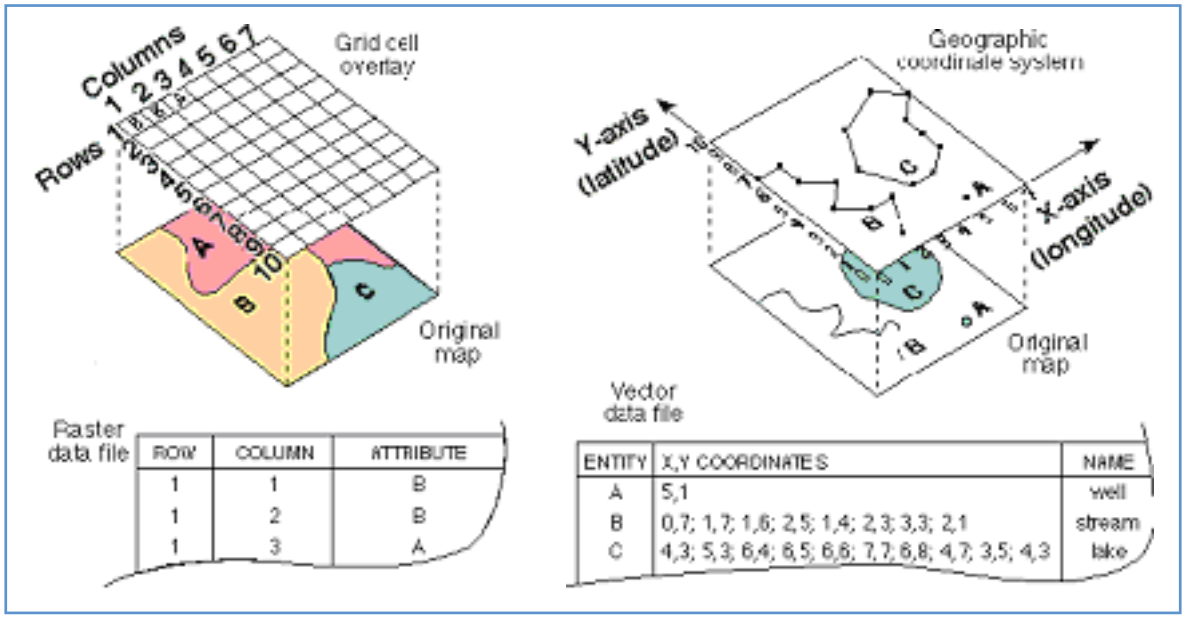



Figure 1, Raster and vector data

Spatial data can be divided into two different categories, raster and vector data (figure 1). Raster data can be seen as a large grid which is placed on top of an area which will be transferred into a GIS. Each cell in the grid will have a value, depending on the entity which is found within the grid cell. Examples of typical raster datasets are imagery of satellites, aerial photographs or other scanned pictures which represent information like height and temperature maps. Vector data uses another method to represent geographical features. Entities are represented as objects in vector data, which consist of points, lines or polygons and have geographical coordinates. The coordinates are geographically projected and position the objects on their location. Examples of typical vector datasets are common topographical maps, road maps or political maps.

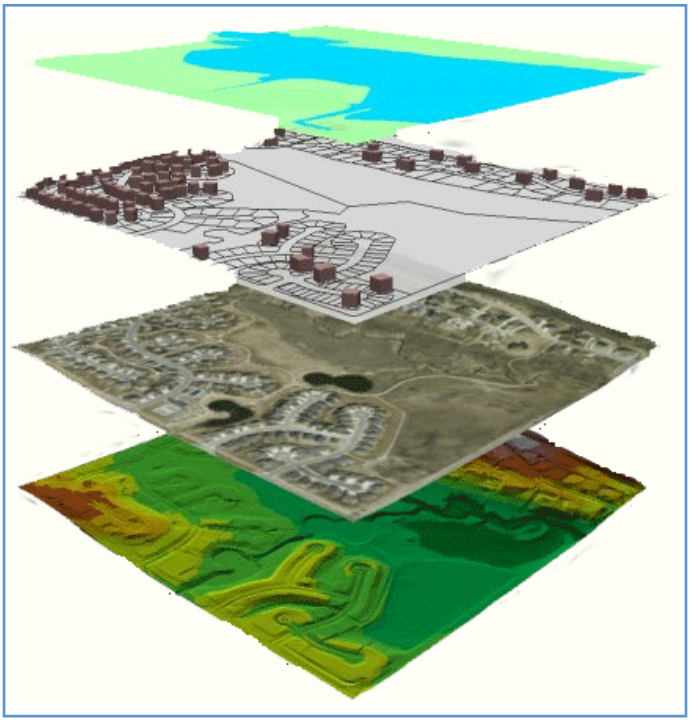


Figure 2, Layers in a GIS

Spatial data is derived from the world around us. Features are commonly represented in a GIS in different layers (figure 2). Each layer contains information related to a specific theme, structures the spatial data, and makes it possible to carry out different analyses and visualizations. Layers can be combined and are used to give a geographic representation, a map or picture of the analyzed phenomenon.

## Challenges in testing a GIS

The complexity of a GIS platform will offer testers different challenges when testing a GIS. Experiences of GIS testing at Alliander has given more insight into some of these challenges testers face [2]. Examples of these problems are: custom code which has been added to the original GIS platform, a comprehensive GIS version management system which supports a multi-user edit environment, and the huge number of possible data variations.

### *Platforms versus custom code*

A GIS usually runs on a platform, examples of these platforms are ESRI or Smallworld. The GIS usually uses standard platform functionalities and resources, but can also run custom code. The combination of standard functionality and custom coded functions makes the GIS platforms very powerful, not only are you able to adjust a GIS to meet your functional demands, you can also tweak the system to run it just the way you want it. This combination also complicates matters, because custom code can conflict with standard platform functionalities and vice versa. This makes it difficult to assess the technical risks that are introduced when implementing custom code. Specialized architectural and programming knowledge is necessary for this risk assessment.

### *Version management*

Because of the strong multi-user character in which a GIS is used at Alliander, version management is a must. Many users are editing a piece of the giant puzzle. Version management is usually standard platform functionality, but when it doesn't function as designed, the consequences can be catastrophic. When errors are made directly in the production data, this can result in damaged gas pipes, electricity cables and telecom cables; it can even result in personal injuries or worse.

Well configured version management makes sure that you are not creating critical errors directly in the production version of the data. Users can create their own editable version of the production data, thus creating a controllable testing environment. In this environment the user can create, read, update, and delete objects while not editing the production data.

When different users alter the same data, it can result in conflicting versions. When checking conflicting versions of the same data, a confliction manager will help to choose the right version. This function is important to ensure sufficient data quality. When confliction management doesn't function properly, conflicting versions of one situation can lead to data corruption and to on-screen errors. Version management is difficult to test because of the coverage issues it contains. Product risks are numerous and need to be taken into account during the product risk analyses.

### *Data variation and topology*

Existing testing methodologies today are specific for testing administration systems. Testers are familiar with testing business processes, workflows and structured information streams. These offer a structured flow of information and actions to be taken in a specific order. There's input, processing and output in an orderly manner. You put in a name, sex, address, birthplace and date of birth and you have successfully registered a person.

A GIS works with spatial data, objects have a specific location and spatial relation with other objects. The number of objects in a GIS and the data variations are enormous. Per object topology rules are configured. These topology and business rules define with which objects it can interact and in what way. In a GIS an object can be placed in numerous orders and connected to many possible objects with many different attributes. Typical work in an editing environment is revision work, changing the registered network according to the applied changes in the field. The editing process in a GIS is not easy to test, because of the spatial dimension and spatial relations objects have. Multiple variations per object or situation are possible, so the coverage of your test should be accordingly.

The number of possible combinations is extremely high. Therefore the usual coverage, which is common for administration systems, does not apply for a GIS. Are all combinations to be tested? How can an efficient coverage be obtained? What product risks can be identified, and how can a complete test coverage be obtained?

## GIS testing, an overview

When using the currently available testing techniques, it doesn't seem they can be put to good use when testing a GIS. The Test Management approach (TMap) lists a variety of testing techniques that aren't sufficient to achieve the test goals when testing a GIS. The simple vastness of data combinations, complexity and the spatial aspect make it difficult to structure the test analysis. From a Business Driven Test Management point of view the following challenges can be identified:

- Identifying product risks in relation to registering objects and additional analysis functionality. It is difficult to tie consequential errors to the object registration. A proper product risk analysis with stakeholders of the object registration process should suffice.
- Matching the appropriate testing techniques with specific test goals. As the current available testing techniques are insufficient, matching an appropriate technique can be difficult.
- As the data variation and topology rules result in an extremely high number of possible test cases, defining the desired coverage and making it transparent is challenging.
- Selection of the right initial situation for test cases is a must to ensure efficient test execution.

With the large amount of testing techniques, it must be possible to relate a certain GIS test goal to a test technique. However, the visual and spatial aspect is not easily converted into test cases with sufficient coverage. The solution is found in combining the quality characteristics of GIS and testing. This results in a viable solution to reduce product risks in GIS development. A quality characteristic can be of more or lesser importance depending on the way a GIS application is used within a company. Based on the quality characteristics which need to be tested one can select the proper testing techniques. The testing method TMap uses quality characteristics based on ISO9126 [3]. These quality characteristics will suffice in most cases. In case of the GIS domain, supplementary GIS quality characteristics can be defined, which can help to structurally test a GIS.

| GIS technique | Topology test | Stylingtest | Cartographic elements | Plottest | Geoprocessing | Raster | Projections | Layers | Versions |
|---|---|---|---|---|---|---|---|---|---|
| **Tmap Next Quality characteristics** | | | ■ | | | | | | ■ |
| Managability | | | | | | | | | ■ |
| Security | | ■ | ■ | ■ | ■ | | ■ | ■ | ■ |
| Usability | | | | | | | | | |
| Connectivity | | | | | | | | | |
| Continuity | | | | | | | | | |
| Controlability | ■ | | | | ■ | | | ■ | ■ |
| Flexibility | | | | | | | | | |
| Functionality | ■ | ■ | ■ | ■ | ■ | | ■ | ■ | |
| User-friendlyness | | ■ | ■ | ■ | ■ | | | ■ | |
| Re-usability | | | | | | | | | |
| Suitability Infrastructure | | | | | | | | | |
| Suitability | | | | | | | | | |
| Maintainability | | | | | ■ | | | ■ | ■ |
| Performance | | | | | ■ | | | | |
| Portability | | | | | | | ■ | | |
| Testability | | | | | | | | ■ | ■ |
| Economy | | | | | | | | | |

| GIS Quality characteristics |
|---|
| Visualization |
| Plotting |
| Geostatistics |
| Networkanalisys |
| Navigation |
| Raster |
| Cartographic |
| Meta-data |

Figure 3 Mapping GIS quality characteristics and techniques

Figure 3 gives an overview of GIS testing techniques and their relation to common and GIS specific functional quality characteristics. Based on these quality characteristics, the right GIS testing techniques can be selected. The visualization quality characteristics can be tested by using a styling test (this will be used as an example later on in this article). The following functional GIS quality characteristics are defined:

- **Visualization** means the broad scale of how a GIS visualizes the data. Is the style of an object correct? Are the right color schemes used?
- **Plotting** means the creation of a map product, physical or digital, within a GIS. It differs from the cartography characteristics because the focus here is on the product itself. Is the styling of the product similar as in the GIS? Are all elements which need to be on the map product correctly filled and placed?
- **Geostatistics** are spatial statistics carried out within a GIS. This is one of the more difficult quality characteristics to test without in-depth knowledge of the subject. Are the static results created by the GIS for that region correct? Are the distortions of the projection and data overlays corrected and taken into account in the statistics calculations?
- **Network Analyses** are a specific kind of spatial analysis which highly rely on correct network logic and topology in the GIS. Is the shortest route really the shortest possible route? Are the calculated driving times correct?
- **Navigation** focuses on how someone can navigate through the geographic data within a GIS. Are the right zooming levels used? Is it possible to correctly browse through the available layers and bookmarks?
- **Raster** comprehends the quality of functionalities within a GIS related to raster data. Are sinks, cells with no data, filled correctly? Are the right color bands used?
- **Cartography** is the study and profession of mapmaking. Related to GIS and testing, it will check if correct visualization techniques are used and if all map elements are correctly implemented.
- **Metadata** gives information about the data used. It will test if all the metadata elements are available and give the right information about the data being used.

### *An example - the styling test*

Before determining the appropriate testing techniques, one needs to determine which quality characteristics are most important and which aspects of a GIS are affected by this quality characteristic. For these quality characteristics specific testing techniques or variations of existing testing techniques can be defined (figure 3). A good example is the visualization of objects. In a GIS visual defects are seldom "cosmetic". All visual aspects are functional and must be styled as designed.

There's a saying: "A picture is worth a thousand words". In testing terms you can put it as: "A picture is worth a thousand test cases" . Because the data in GIS is visualized, a single picture can represent numerous pages of alphanumeric data. The information density of the visualized data is very high. In a GIS the visualization is managed by configuration for styling, visibility rules and user settings. These visualization rules are complex and it's easy for a software developer or configuration manager to introduce a bug. These visualizations are common to be different for each specific scale. A GIS must be able to convert this data into clear and understandable images. But how do you define "clear" and "understandable"?

One example of a GIS testing technique which can be used to test the visualization quality characteristic is the styling test technique. The styling test is based on lessons from cartography and will look if the right styling method on an object is used and if data is correctly visualized. It will use multiple testing techniques which form a structured approach to test styling. There are some general aspects which together determine for the most part if a feature is correctly styled.

The styling test defines conditions which are used to test the styling characteristics of an object. In this case the scope is limited to the position characteristic, normally other styling characteristics should also be integrated to run the styling test. The first step is to determine which of these aspects are important for which objects and results in an overview of the aspects and objects which need to be tested. The second step is to create a decision matrix. In this matrix an overview of possible test cases is given. In the final step test cases will be further detailed.

### *Step one: Define styling coverage*

The first step is to select for each object type which styling characteristics are more important and to what extent they need to be tested. These styling characteristics are specifically related to a GIS. All of these styling characteristics have a different manifestation depending on the data type, this can be raster or vector data. In case of vector data it can be a point, line, polygon or volume (three-dimensional). Depending on the coverage needed to test a characteristic of an object, different variations need to be integrated in the test case. Figure 4 gives an overview of the primary styling characteristics and an example of how they can be related to object types. The bullets indicate the required test coverage.

| | Gas Value | Gas Pipe | Pipe labels |
|---|---|---|---|
| Shape | - | - | - |
| Color | • | • • | - |
| Gray value | - | - | - |
| Intensity | • | • | - |
| Symbology | - | - | - |
| Positioning | • • • | - | - |
| Readability | • | • | • |
| Visual hierarchy | • | • • • | • • |

Figure 4

- ***Shape*** is the correct absolute data visualized, are the proportions of the symbol correct, isn't it too tall, too broad, too small etc.?
- ***Color*** is the correct relative data visualized, are the correct colors used, are these cartographically correct?
- ***Gray value***, are proper gray values being used?
- ***Intensity***, is the contrast of the image correct, don't they stand out too much?
- ***Symbology***, is the right symbology chosen, does it have the right association or resemblance, is it conform to a standard or convention?
- ***Positioning***, are the symbols placed correctly, is the rotation correct?
- ***Readability***, is the symbol clear enough, are labels readable, are the correct labels and values displayed and don't they overlap?

- ***Visual hierarchy***, are the right symbols drawn on top of the others? Are all objects within a picture properly visualized?

Each of these primary styling characteristics is related to secondary styling characteristics. Depending on the needed test coverage these secondary characteristics will help to determine appropriate test cases. The positioning styling characteristic can be made up by secondary characteristics, for example: rotation, relative location, and alignment.

- ***Rotation***, the object will have a direction or rotation. The rotation can be tested by checking if a rotation is carried out correctly Different rotations can be used in a test case.
- ***Relative location***, in most cases the visualized symbol will be larger than the exact location. The relative location determines where a symbol needs to be drawn. It could, for example, be at the center of the symbol or at the lower left corner.
- ***Alignment***, in some cases objects need to be aligned properly, for example an object can be aligned in relation to other features.

In this case the object will have a rotation depending on the gas pipe at which it is drawn. Because of the high test coverage, this needs to be tested for rotations of 0, 90, 180, and 270 degrees. The xy-coordinates will always be at the center of the symbol. The object must be aligned to the gas pipe underneath. The relative location and alignment will implicitly be tested when testing the rotation. Figure 5 gives an overview of the position conditions which should be tested with illustrated examples.

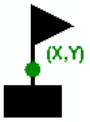
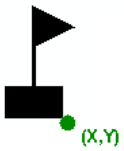
| **Positioning** | | | | |
|---|---|---|---|---|
| Rotation | 0°, 360°  | 90°  | 180°  | 270°  |
| Relative location | South South   | Centre Centre   | South East   | South West   |
| Alignment | Aligned pipe 1  | Aligned pipe 2  | | |

Figure 5, Positioning characteristics

### *Step two: Integrated decision table*

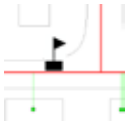
The object will be visualized depending on certain settings in a GIS; a decision table can be used to design test cases. In this case the object should always be visualized if both the visibility and the discipline are turned on. If it is highlighted it should also be displayed. This decision table will be integrated with the previous position conditions which need to be met (figure 6).

| **Gas Value** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Visibility on | Y | Y | Y | Y | N | N | N | N |
| Discipline on | | | N | N | | | N | N |
| Highlight on | | N | | N | | N | | N |
| **Position** | | | | | | | | |
| Rotation | 0 | 90 | 180 | | 270 | | 0 | |
| Relative location | \| | \| | \| | | \| | | \| | |
| Alignment | \| | \| | \| | | \| | | \| | |

Figure 6, Gas valve integrated decision table

### *Step three: Test cases*

In the final step the test cases needs to be defined which are used to test if the object complies to the previous conditions. This can either be done by using specific test data or production data. The expected result which meets the conditions and location parameters will be integrated in the test case. Once finished the tester could easily check the right visualization settings, navigate to the location and check the GIS result compared with the expected result (figure 7).

| Gas Value | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Visibility on | Y | Y | Y | Y | N | N | N | N |
| Discipline on | Y | Y | N | N | Y | Y | N | N |
| Highlight on | Y | N | Y | N | Y | N | Y | N |
| **Location** | (322, 432) | (422, 332) | (376, 372) | (376, 372) | (314, 432) | (314, 432) | (222, 414) | (222, 414) |
| **Result** |  |  |  | None |  | None |  | None |

## Conclusion

The example showed how a GIS quality characteristic can be related to a GIS testing technique. In this case the styling test was used to test the visualization quality characteristic. The technique uses several steps to structurally approach the quality attribute by defining related characteristics which can be used depending on the needed test coverage. The final step results in specific test cases and expected results which cover the styling characteristics defined earlier.

The spatial component of a GIS makes it difficult to structurally test a GIS. The problems involved when implementing custom code, the huge data variations and extensive version management system are just a few examples of problems encountered when testing a GIS. Implementing existing testing techniques will result in thousands of test cases. The GIS testing techniques which are developed by Alliander and Bartosz are a combination of existing and new techniques which will give testers the tools to structurally test a GIS. Although the example uses only one piece of an extensive GIS testing technique, it does give an indication of how GIS quality characteristics can be narrowed down to specific GIS test cases. Although structurally and efficiently testing a GIS might at first seem impossible, it can be done with adjusted and new testing techniques.

## > biography

**Harmen de Boer**
*is a test manager at Bartosz ICT in Arnhem, The Netherlands. He studied corporate informatics at the Informatics and Communication Academy in Arnhem. With six years in testing, the last two years were spent on a challenging assignment at Alliander. His testing experience was mostly obtained in the finance and industry sectors. In these assignments Harmen gathered experience in test analysis, test coordination and test management. The GIS domain offers Harmen new challenges in structured testing and test management. Together with Jeroen van der Zwan, new approaches and techniques are being developed to structurally test a GIS.*

**Jeroen van der Zwan**
*currently works as a GIS functionality manager at Alliander, The Netherlands. After his studies of Human Geography, with emphasis on GIS, at the University of Utrecht and National University of Singapore, he started working as an officer geospatial analyst at the Royal Netherlands Army. He continued his career as a Geo-ICT consultant at Logica and worked for 2 years at Alliander on GIS and testing. As from September 2010, he joined Alliander continuing his work in the GIS domain.*


# The Domain Risk Analysis: Taking risks to a higher level

*by Chris Schotanus & Bart Knaack*

In modern test (management) methods product risks are taken as input for determining priorities in testing. Some methods also report back in terms of these risks, so that it is easier to decide whether to go live or not. This means that each project will start with a product risk assessment involving several stakeholders.

Although these sessions render valuable input for the testing process, they are not always done in the most efficient way. Stakeholders that are part of multiple risk assessment sessions see the same risks being raised over and over again and start to lose interest. As a result the product risk assessments turn into a ritual show that needs to be run at every project.

## Product and project risks

Risk management plays an important role in daily management. Within a test project there are two types of risk: test project risks and product risks. Test project risks refer to situations that may endanger the test project. Product risks are the risks that arise from the use of a system by a company or organization. Insight into product risks is important where decisions must be made to take an information system into production. In this article, we lay the emphasis on the product risks.

## The product risk analysis

Stakeholders are all those people and/or departments that have a special interest in the correct functioning of an information system. They are the people for whom the test project is being conducted. That's why, to reach the most accurate risk analysis, stakeholders' input is essential.

After identification of the product risks, the priorities within the set of product risks are thought about. Once determined, the test effort can be focused on those parts of the system that pose the highest risks. We use the product risks as a means to manage the test project; based on the priorities assigned by the stakeholders. the scope and depth of the test can be defined. The requirements that cover the highest risks are tested first, those with lower risks are tested later or not at all. Also, the report by the test manager is based on these priorities. In the intermediate and final reports the test manager will indicate which product risks are treated correctly and which are still open, which will help the stakeholders decide whether to accept.

## Current process PRA

The current methods for risk based testing advocate that the product risk analysis should be done at the start of the project. The stakeholders and test manager should perform a product risk analysis together. This product risk analysis may take place in different ways. In many cases the stakeholders will be interviewed individually. While this is an accepted method which may yield substantial results, it's very time-consuming.

The statements of one stakeholder may influence earlier statements of other stakeholders. This means additional interviews. An approach that's considered more efficient is the Product Risk Analysis Workshop (PRAW). During a PRAW the stakeholders jointly identify the product risks. It's supervised by a moderator to ensure things go smoothly. This workshop can use various techniques like a brown-paper session, a Failure Mode Effect Analysis (FMEA) or a Mind Map.

In these sessions we typically also want to prioritize the risks and aggregate / split them up to an equal level of detail such that they can be rated in severity and can indeed be used appropriately for risk mitigation actions. All this is a lot of work, but when done properly it really helps to direct the entire testing effort in the right direction.

The quality of the output of such a session, however, is dependent on:

- The group of stakeholders present (have we identified the right group?)
- The understanding of the goal of the PRA session by its contributors
- The capacity of the facilitator (and the group) to distinguish between project and product risks
- Domain knowledge of contributors
- Legal/regulatory and standardization knowledge
- The capacity of the group to take each other's risks seriously
- The commitment of the contributors
- The duration of the process.

# Knowledge Transfer – The Trainer Excellence Guild

Díaz Hilterscheid

## Rapid Software Testing
by Michael Bolton

- May 23 – 25, 2011 Helsinki

## An Agile Approach to Program Management
by Johanna Rothman

- Sep 12 – 13, 2011 Berlin
- Sep 15 – 16, 2011 Amsterdam

## Foundations of Agile Software Development
by Jennitta Andrea

- Jul 7 – 8, 2011 Berlin
- Jul 11 – 12, 2011 Amsterdam

## Risk-Based Testing
by Hans Schaefer

- Jun 10, 2011 Kopenhagen

© iStockphoto.com/oxideluxe

**Website:**
www.testingexperience.com/knowledge_transfer.php

As mentioned, the process of risk elicitation can be very time consuming and facilitating such a session is not trivial. Since the outcome of a risk assessment is dependent on many factors, many do not produce the intended result. Since this risk list is used to steer the testing activities, a 'defective' risk list will produce an ineffective test process

## The domain product risk analysis

If we look at the product risks in more detail, we see that several product risk assessments within the same domain will lead to similar outcomes, yet small differences occur based on the specific project at hand. It seems that there is a lot of duplication of effort.

However, if we turn to analyzing the product risks at domain level and hand the results down to any project being started within that domain, we can use the predefined risk list as a guideline and determine what specific risks we might need to add and what weight to attribute to the different risks. In doing so we are less liable to forget important risks or starve our resources by doing too much redundant work. Furthermore, the repetitive, and in the end more or less boring work is taken away which will speed up the product risk analysis process.

## How to do a domain risk analysis

When we are going to do a domain risk analysis, we start to look at a bigger picture and try to produce a result that is reusable. Investing additional energy to get this analysis right will therefore pay itself back through the fact that it is reused and kept up-to-date by subsequent projects. We might therefore split the domain risk in a number of sessions and try to get the best representatives for each stakeholder to participate preferably at program level instead of project level. Since the outcome of the session is highly dependent on the capability of the facilitator, we also need to have the top dog of this pack.

In many ways running a domain risk analysis session resembles a product risk analysis session. Since we are, however, no longer dealing with a single project with a specific impact, we need to add some steps at the beginning of the process:

- Identify the critical chains within the domain or impacted by the domain
- Identify the critical activities/tasks with your domain
- Identify the critical business processes supported by your domain

Afterwards we perform a 'Product risk analysis' session based on these activities and processes, relating the risks to the activities, processes and chains that you have identified. Although this sounds simple enough, it does require the contributors to reflect in a more abstract way on their work.

Since we perform the domain risk analysis to come to a kind of 'super product risk assessment', we can make use of some best-in-class product risk assessments of projects that have been done in this domain, if they are available

## Applying the domain risk analysis in your project

If the domain risks have been identified, they still need to be applied as soon as a project is started. To do this we identify the following steps:

- Identify what domain(s) your project is part of
- Identify what critical chains/activities/business processes your project impacts
- Copy the risks associated with these from the domain risk assessments
- Evaluate this risk list and take it as a basis for the risk analysis

Once you have this initial risk list, you have to tailor it to the specifics of your project. To do this there are three steps to be taken:

- Identify whether the project introduces new critical activities to the domain
- Add new product risks induced by the critical success factor of your project
- Investigate whether the technical complexity of the (addition to the) system justifies a reclassification of the risk impacts

Depending on the size of the project and or the impact of the change, these questions can be dealt with by a desk session, or still require a product risk analysis session. However, since most of the work has already been done in the domain risk sessions, tailoring the risk list based on the considerations above is much easier and less time consuming.

## The need for maintenance of the DPRA risk list

As soon as the domain risk analysis is finished and the first domain risk list has been released, the risk of going out of date arises. After all, product risks and priorities may change over time. Therefore the risk list needs to be maintained. Preferably, a department assigned for maintenance of test products takes up this activity.

The input for this maintenance process will come from feedback from the projects.After each product risk analysis done at the start of a project, the results of it are sent to the maintenance department. Here the risk list is compared with the domain risk list. Each deviation is then discussed with the stakeholders and, if there's any need, the domain risk list is adjusted.

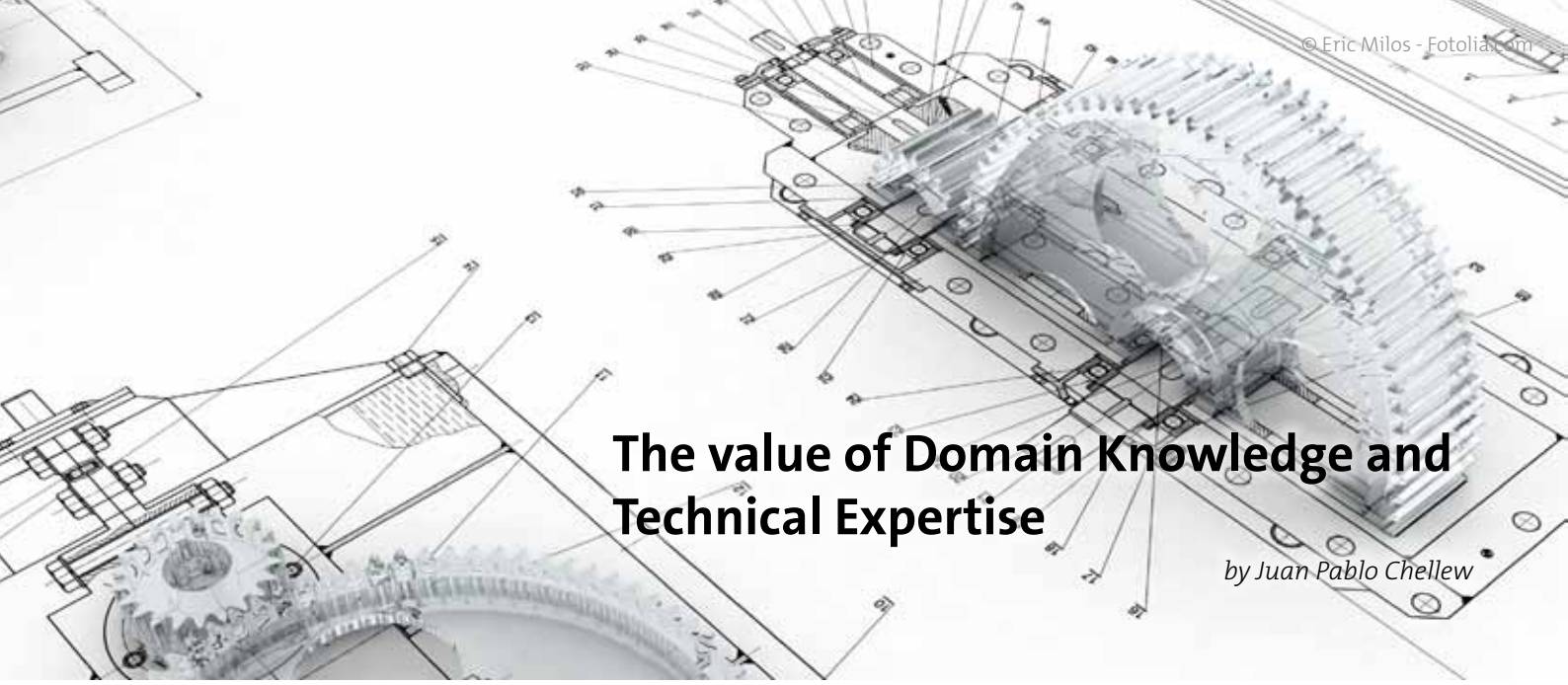
## Conclusion

The product risk analysis and the product risk list produced during the process are of vital importance to a test project. It allows the stakeholders to define acceptance criteria and it supports the test manager in defining the scope and in planning the project. From experience we know that the product risk analysis within a particular domain often leads to the same product risk list. By defining the product risks for that domain, the time spent on performing a product risk analysis in a project will decline, which will improve the efficiency of the test process.

## > biography

**Chris C. Schotanus**
*has more than thirty years of experience in IT, more than fifteen of which were spent in the field of testing, test management, setting up test organizations and test automation. As a principal test consultant for Logica in the Netherlands, he advises organizations regarding test policy and test organization, both in the Netherlands and internationally. He is the co-author of the books 'TestGrip: Gaining Control of IT Quality and Processes through Test Policy and Test Organization' (2007) and 'TestFrame, a method for structured testing (2008)'. Contributions by him can also be found in 'Successful Test Management: An Integral Approach (2004)' and several other books and publications on testing.*

**Bart Th. Knaack**
*is senior test advisor at Logica. He has a broad understanding of the entire software development life cycle and has contributed to IT projects in many different roles, such as developer, development lead, quality assurance officer, tester, test manager, test automator and test advisor. For the last 15 years he has focused his attention on testing. He has spoken at many testing conferences such as Eurostar and JTS2009. As co-founder of the Eurostar testlab he is an active contributor to knowledge activation in the testing domain.*




# The value of Domain Knowledge and Technical Expertise

*by Juan Pablo Chellew*

When we talk about *domains* this has nothing to do with domain names, domains in the Windows networking world, or mathematical uses of the word "domain". We are referring to businesses or specific industries.

Let me start by defining these two areas. *Domain knowledge* is the area of expertise related to the environment of the application, whereas *technical expertise* is the technology used to build the application and the knowledge or know-how to execute day-to-day activities to get your work done. Based on these terminologies, examples for domain knowledge would be mobile, VoIP, banking, e-commerce or health care applications and technical expertise would be protocols, networking, programming languages, web, search engines, SOAP, testing and automation, to name a few.

Throughout my QA career I have always looked for new hires with testing experience, some technology knowledge such as Oracle, SQL, Linux, basic programming skills, web, client server and some domain knowledge associated with the industry related to the work environment. Looking back through my career I had analyzed my personal experiences and realized that the market domain was less important than my technical expertise. Through this analysis I have come to the conclusion that technical knowledge is more valuable and sellable than domain knowledge. This is especially true if you are looking for an entry level position; it is extremely rare to find an entry level individual that has domain knowledge, and new graduates will always have the technical knowledge as this is taught in college. At the very least, they must be highly familiar with how software works as well as how it is assembled. They must also have an extensive knowledge of computers, knowledge of testing principles, and good people skills or communications skills.

Through the years I have noticed that the responsibility of a software tester is to evaluate the system's functionality based on requirements or business scenarios and to document the deviations of such in a detailed and reproducible manner. 90% of a test engineer's work is related to technical knowledge and the other 10% may be related to the domain knowledge, i.e. how a system behaves when it is configured specifically for a given industry. Domain knowledge comes in handy during design reviews and perhaps in the documentation of test cases. However, when the real testing begins and the test engineer rolls up his/her sleeves, it is the technical knowledge that counts. The test engineer must be familiar and understand the complexity and the specific intricacies of how the application was built as well as the complexities of the environment and its interactions with other software or applications.

A test engineer must know his or her way around the technology; it is just not a matter of running test cases to comply with the requirements, coverage and reporting abnormalities or deviations of the requirements; they must dig much deeper than this. Once an issue is identified, a more in-depth analysis needs to be conducted to provide as much detail and knowledge of why the abnormality has occurred. Where did the issue originate? In the client side or front-end, middle layer or back-end? Here is where the tester's technical knowhow and experience becomes significant.

I would agree that some scenarios exist where the domain knowledge will have a great advantage over the technical expertise, i.e. in cases where testing is conducted without proper requirements, use cases or business scenarios. However in today's world these key documents are a must in order to produce quality software. The domain knowledge will help you at the beginning of the cycle, you might even be able to contribute to the design of the application, make usability recommendations and possibly assist in the test case design, but the domain knowledge will not really help you in finding critical defects. Domain knowledge is the beginning of testing knowledge. I consider the domain knowledge to be the first layer to the success factor of a test engineer, and as you get more involved in testing the technical skills will be at the core of your success.

I see business analysts, product managers and software architects needing more domain expertise than technical knowledge, as these are the people designing the system for a particular industry and passing down their domain knowledge through the requirements, use cases or business scenarios to the rest of the team.

Some people believe that it is easier to train someone in the testing process than in a specific domain. In my experience, however, it has been the other way around. If a test engineer knows what to look for in the form of defects, how defects could be crea-

## Online Training

Díaz Hilterscheid

**ISTQB® Certified Tester Foundation Level** (English & German)
**ISTQB® Certified Tester Advanced Level - Test Manager** (English)
**ISTQB® Certified Tester Advanced Level - Test Analyst** (English)
**ISTQB® Certified Tester Advanced Level - Technical Test Analyst** (English)
**ISEB Intermediate Certificate in Software Testing** (English)



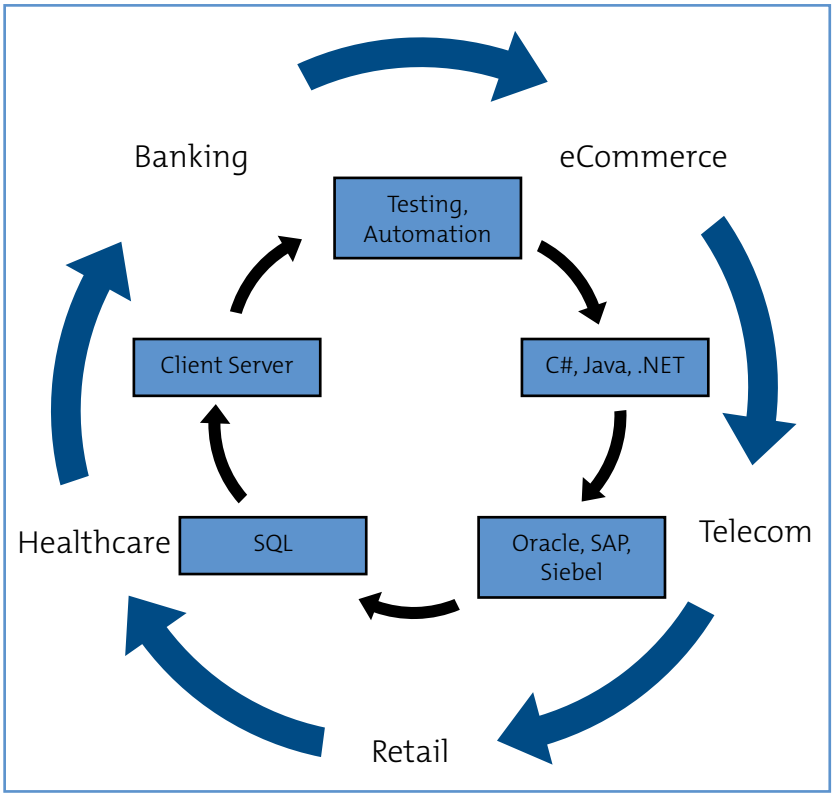

iseb Qualifications from BCS

ISTQB® Accredited Training Provider

ISTQB® International Software Testing Qualifications Board

Díaz Hilterscheid

te testing experience



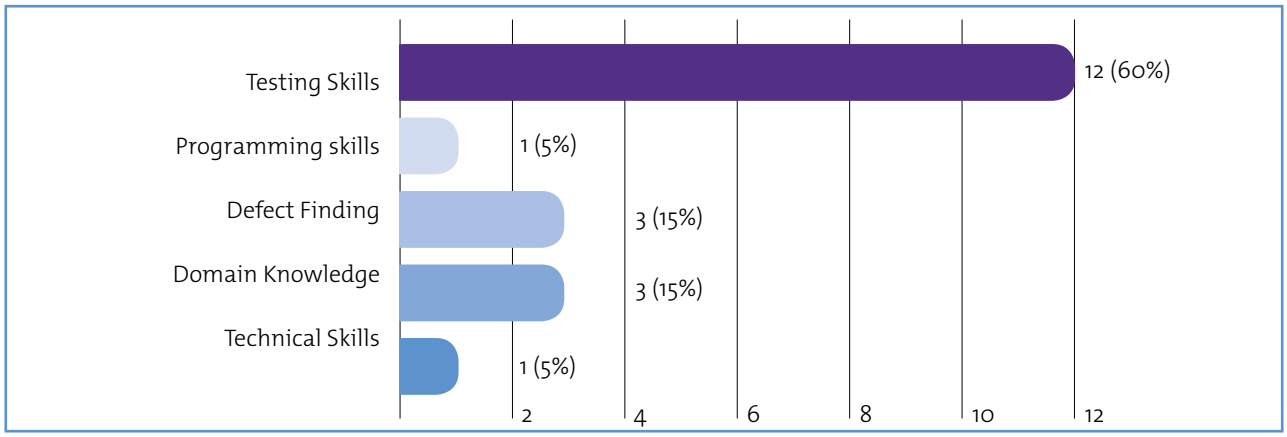

Layers of a great tester

ted, the potential impacts of code modifications and technology impacts based on languages or architecture, this is more valuable than knowing what the application will be used for during testing. If you were to look for a new programmer you would first see if he had the technical expertise, such as programming language, and then you would look to see if he had the domain knowledge. If you browse through software development job postings, the domain is rarely a requirement and almost never mentioned. In some cases the domain is irrelevant as long as he/she has the technical know-how.

When I evaluate the level of a test engineer, I basically consider three categories: 1) testing skills, 2) technical expertise, and 3) domain knowledge. Categories 1 and 2 can be taught in a class room or online courses, Category 3 can only be taught or learnt through experience, and each domain knowledge is particular not only to the industry but to each corporation. Companies tend to modify their processes and modify domain standards to meet their needs. Even though there are domain standards just as there are programming standards, they vary based on market and customer demands.

In a poll conducted where I asked a group of people in the software testing industry what they considered to be more important in the success of their career, the majority of the respondents stated that having testing skills is more important, next came defect finding followed by domain knowledge. I consider testing skills to be part of the technical skills required for a good software tester.

In another poll respondents were asked to select what they consider to be more important and valuable for a test engineer: domain knowledge or technical expertise. The answers were almost split in half where 51% of the respondents opted for domain knowledge and 49% selected technical expertise. 90% of the respondents do believe that it is easier to learn the domain and that, depending on the level of testing you are performing, each domain and technical knowledge has its place in being a successful test engineer. Most respondents agree that if you are involved in User Acceptance Testing (UAT) or system level testing, the domain knowledge is a "must-have", and if you are doing unit or functional testing then technical expertise is more valuable. Most of the respondents agree that both domain knowledge and technical skills are essential and that you cannot succeed with just one and without the other.




Poll 1: Key Success Factors for Software Test Engineers

In my situation it has been easier to transition from one domain to another, but it has taken me longer to absorb and learn new technologies, especially when you move up the ladder. I have successfully been able to transition my technical expertise from one domain to another. The technologies, such as SQL, programming languages, testing techniques, CRM, ERP and others, are present across domains.

The testing aspects or technical knowledge have been harder for me to learn due to the fact that there are many areas in this overall subject in which a tester will need to have expertise, including testing skills, defect identification skills, communications skills, automation skills, programming and the ability to work under pressure. All of these technical skills are constantly changing and, as I see it, these changes contribute to the changes in the domains.

In today's world you can observe that the professionals that are hired in different companies tend to be those with domain expertise rather than those having technical skills. Currently the software industry is also seeing a positive trend that many professional developers and domain experts are moving into software testing. Even if you know your domain, you would not know where to look for defects or what could be the potential outcome of a particular fix.

The flow of data in a system is domain independent. Data, for example, is collected (manually or automatically), validated, stored in a database, manipulated (updated or deleted), and then displayed back to the user or used in some sort of report. These processes happen throughout all systems, irrespective of what the business domain is.

No matter what domain you find yourself in, you will always need the technical knowledge. A test engineer can thrive and be successful with little domain knowledge if he expands his/her technical knowledge. If you are a domain expert outside development, you will not need a lot of technical knowledge. However, if you are a technical expert, you will need to familiar with basic domain knowledge in order to be successful in software testing.

## > biography

**Juan Pablo Chellew**

*Aristotle once said, "The greatest virtues are those which are most useful to other persons". I believe that your biggest contribution to society is passing your knowledge on to others. Juan Pablo Chellew has been leading QA departments for over 17 years and is currently QA manager for RedPrairie. RedPrairie delivers productivity solutions to help companies around the world in three categories – workforce, inventory and transportation. Pablo has worked in the field for more than 17 years. He has experience leading the efforts in all areas of software testing including test automation, performance testing, usability, process improvements through lean development, and software testing methodologies such as Waterfall, A.I.M., RUP, and Agile (Scrum). Pablo's experience involved the following domains: telecom, e-commerce, retail, communications, IT and COTS.*

# Can agile be certified?

Find out what Aitor, Erik or Nitin think about the certification at **www.agile-tester.org**

## Training Concept

All Days: Daily Scrum and Soft Skills Assessment

Day 1: History and Terminology: Agile Manifesto, Principles and Methods

Day 2: Planning and Requirements

Day 3: Testing and Retrospectives

Day 4: Test Driven Development, Test Automation and Non-Functional

Day 5: Practical Assessment and Written Exam

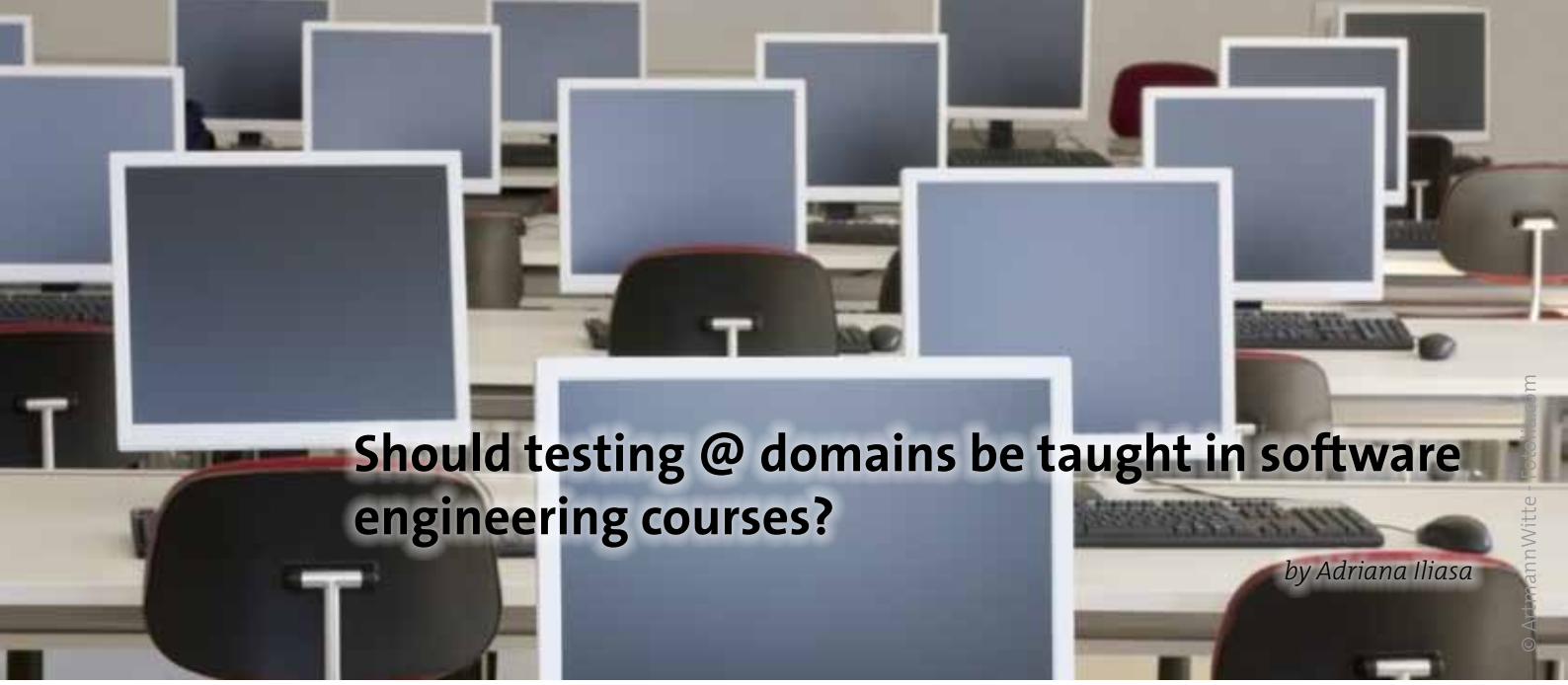
# Certified Agile Tester

We are well aware that agile team members shy away from standardized trainings and exams as they seem to be opposing the agile philosophy. However, agile projects are no free agents; they need structure and discipline as well as a common language and methods. Since the individuals in a team are the key element of agile projects, they heavily rely on a consensus on their daily work methods to be successful.

All the above was considered during the long and careful process of developing a certification framework that is agile and not static. The exam to certify the tester also had to capture the essential skills for agile cooperation. Hence a whole new approach was developed together with the experienced input of a number of renowned industry partners.

Supported by

Barclays
DORMA
Hewlett Packard
IBM
IVV
Logic Studio
Microfocus
Microsoft
Mobile.de
Nokia
NTS
Océ
SAP
Sogeti
SWIFT
T-Systems Multimedia Solutions
XING
Zurich



# Should testing @ domains be taught in software engineering courses?

*by Adriana Iliasa*

Googling around the Internet I noticed that more and more articles appear complaining about the lack of preparation of new graduates. Usually these complaints refer to their formation as experts: they only have a general idea about their specialization, and there are often cases when they are not even able adapt to new situations.

Companies say that it is not cost effective to hire new graduates or students, and to then teach them everything. And they expect universities to carry the burden of forming experts.

Universities, on the other hand, respond that they don't have enough time and money to prepare good and enthusiastic professionals.

As a recent graduate I started wondering how this situation can affect me when I recently entered the field of software testing, and of course I wanted to do my best.

Being the type of person that likes to get involved and understand what's happening around me and the impact that the used software and technologies have on everyday's life, I did a little research amongst my colleagues and friends. I wanted to know how aware they were about the software that surrounds them.

I was rather unpleasantly surprised to find that their attitude was one of "I blindly trust the computer. As it can never go wrong, I don't care...". Bearing in mind that many of them had gone to software engineering classes with me, this made me realize that the situation is worse than I thought.

I started wondering: Should I blame the students for their lack of interest? Should I be upset with the professors, or with the curricula? Or should I blame the economical-political system?

When an airbag suddenly opens, without any reason, putting the life of the driver in danger, or when modern medical equipment isn't accurate, we start asking questions similar to the ones above. When accidents happen because of software errors, we tend to find the guilty ones.

However, if we think it over, that is not the right attitude. It's just like in medicine: If you cure only the effects and not the cause of the illness, the disease will come back over and over.

Software engineers and especially testers should be very aware of the importance of the domain in which they are testing, and that should happen before they make a mistake. It is one thing to test software that sends signals in case of a car crash, and it's a totally different thing to test the software of a bank.

The tester must understand the business behind it, and also how failure will impact society. He must imagine situations in which the program may not behave correctly, and must identify and insist on those scenarios that may cause damage to the user.

As a student I noticed that software engineering courses do not highlight the importance of software testing at domains. Professors tend to move right away to unit testing and other programming issues, probably because the design and test planning stage is considered trivial. Their teaching about the importance of the domain in which you test reduces to "Well, if you have an airbag launcher application, you must test it right, or else many people may be injured". What "test it right" actually means, however, remains a mystery.

This lack of information may be thought as excusable - too little time to go into details. However, I believe that actually this lack of awareness of not knowing when, how and what to test results into so many failed projects.

There are places where the collaboration of local companies with the faculties is encouraged, in order for students to get their necessary experience. I believe that is a great idea, and it should deepen in the years to come.

Nobody and at the same time everybody is responsible for bad software. Good software is actually well tested software, and that can be done only by a well prepared tester in that domain (whether it is the automotive, business, or health and pharmaceutical industry). And you can't get a good tester if he/she doesn't spend time also studying that part of the industry.

As students and professors say: “It is impossible to learn about everything in school.” There is no time, sometimes no finances, and a school can’t simulate the environment of the future company in which the student will work.

That is why I believe a strong collaboration between schools and companies should exist, and it should also be sustained by the state. Theoretical knowledge could then be applied in practical hours at the company site, and the student will get a clear understanding of the whole process, will gain experience, and confidence. This way everybody will win something: universities get support and are kept up to date with the new changes and requirements of the industry, students will have a better employment chance, companies will get better productivity and - hopefully - better software will appear to make our lives more comfortable and safe.



## > biography

**Adriana Iliasa**
*I am a junior tester at Infobest and also a Masters student in Statistical Methods for clinical trials in Timisoara, Romania.*


# Importance of domain knowledge in testing applications

*by Krishan Upreti*

***What is Domain Knowledge? Is it something really important for a tester to possess to be successful? Why do clients prefer to have a tester with domain knowledge of their domain? All these questions usually bother all of us testers at some point in time during our careers...***

Domain knowledge is the knowledge/expertise about the environment, system and business processes for which the application is developed and implemented. For instance, when developing an application for food chain restaurants, experienced people from the restaurant industry with domain knowledge about restaurant operations will have major inputs in specifying the requirements for the application. These subject matter experts (domain experts) will be of utmost importance in providing expert advice while developing and implementing the application(s) and for the success of the project.

Domain knowledge is not only an edge to a tester, but it provides an edge to every team member who is part of project development/delivery/maintenance etc. The business analyst, who is part of the requirement gathering team, must be competent in the domain for which the requirements are being gathered, because this project phase is very important. In this phase, the experts need to set and finalize the requirements in contrast to the latest technology/product stack/best industry standards etc.

Having domain knowledge is always an added value for any tester. Domain knowledge not only gives an edge to a tester to uncover the defects which can be found when an application is used in real time by end users, but also provides the edge for impact/defect analysis, for writing defects with more detailed information etc. This also helps tester(s) in identifying more relevant defects because they understand the environment, product technology stack and business processes well.

So, certainly domain knowledge is important to be successful in specific domain testing, e.g. protocol testing, device drivers testing, BFSI domain testing, retail domain testing, healthcare domain testing.

A tester without domain knowledge can find most of the obvious defects in an application, but he/she will not be able to uncover the defects which an end-user of the application might face?

### *Why do customers look for testers with domain experience?*

Today most companies do not look too much for technical testers, but rather for testers with experience in the domain of the application to be tested. This will help testing teams to identify and mimic the real-time scenarios with these domain expert testers. Testing is the last resort for any project team before they move their application in production, so every team wants to test their application in an environment as close as possible to that of production and with scenarios that will happen in real life. The demand for domain expert testers is increasing day by day, because no-one wants to take the risk of taking their application live without having been tested by domain experts. Any critical issues in the business line of the system found after production rollout of the application can result in huge losses to the company, so everyone wants to invest early in resources (i.e. domain experts) who can help reduce these risks and provide quality systems.

### *How does domain knowledge help testers?*

One of the world's biggest fast food chains planned to deploy the learning management system for their managers and crew members at restaurants. A project team finalized the web based learning management system which will be deployed on product company hosted servers accessed over the web by all managers and crew members at restaurants to take up their learning plans. No one in the team ever thought that this application could impact their business. However, during the testing a tester with domain knowledge about retail chains and who understands the business model of the retail industry revealed a test scenario which could have incurred huge losses to the company if this had been missed. This web based learning management system was supposed to be accessed from the machines which are the backbone of restaurant operations including cashless transaction processing. There were a few flash based learning items which were chocking the network bandwidth of these machines responsible for approving the cashless transactions. However, there was no issue in accessing these learning items on the learning management system. The team then realized that this was a critical issue, because if cash-less transaction approval takes lot of time it can lead to huge queues in the restaurant during peak hours which in turn can impact on sales/profit. Finally, this issue got

resolved by providing the light-weight learning items by a vendor who developed these learning items. This is how domain knowledge provides an edge over those testers who just have technical knowledge about the system and do not possess the business domain knowledge.

Points to consider while preparing a test strategy for retail domain applications

1. Consider the different versions of the product/hardware technology stack
2. Consider the business trends, like peak sale hours, peak sale seasons, products with maximum sale etc.
3. Consider the network topology for performance testing
4. Usability testing is very important because any extra minute spent for order processing can lead to huge cash losses and hassle faced by customers at on-line sites when ordering the product, and may move dissatisfied customers to other vendors
5. Consider the web trends etc

***Conclusion:***

Domain knowledge of a tester is very important in delivering the quality products to the market according to market needs. The need for domain specific testers is increasing day by day. So, the more value you provide to customers, the more money you will take home.

## > biography

**Krishan Upreti**

*I am a qualified Master in Computer Application and application cracker by profession. I started my career in the IT Industry around 7 years back and have since then tested and successfully moved many applications for fortune 500 companies into production with minimal issues. No application can be defect-free, but minimizing the risk for business is my job. I have almost 5 years of testing experience in the Enterprise Content Management area with the world's top healthcare and pharmaceutical clients and 2 Years of experience in the Learning Management area with the world's biggest fast food retail chain. Throughout these 7 years of experience I have focused on understanding the customer business, technology as well as Enterprise Content Management and Learning Management products to a great extent. This domain product/business domain knowledge gained over the years helped me in providing high quality deliverables to my clients and gaining an extremely valuable image with my clients.*

# Testing @ Domains

*by Girish R Valluri*

Imagine two people trying to converse and one person does not understand the context. The chances of the conversation being effective are remote. Similarly, if a tester does not understand the domain, software testing can be performed, but it will not be effective and the whole process will not create the level of confidence in the stakeholders that would have been created otherwise.

The importance of domain knowledge in testing increases may times when we consider outsourcing and crowd-sourcing where the testing team is usually not co-located and comes from varied domains.

Let us look at some of the advantages of domain focused testing in more detail:

***Requirements:***

If a tester has expertise in a particular domain, understanding requirements requires less time and the quality of queries for clarification will also be high. The queries / defects raised would help reduce ambiguity in requirements and improve the overall quality of the requirements, thus ensuring that there are minimal iterations of updates to the documents and keeping all parties on the same page. During the requirement phase, static testing plays an important role to identify defects early in the software development lifecycle well before development begins. Static testing is only possible if the tester has domain expertise to understand the customer requirement and the system under test. Hence domain based testing helps to achieve early testing.

***Test planning:***

As a direct offshoot of clear requirements, there is a higher likelihood of effective test planning deliverables such as test scenarios and test cases. In addition, if the tester is able to understand implicit requirements, it will lead to better coverage and less rework.

***Test data management:***

The quality of test data is one of the key factors for the success of the test. Tester with domain expertise will be able to generate more logical test data sets for test execution. Domain experts will be able to derive correct combinations of data for positive as well as negative scenario testing.

***Test execution:***

Applying risk based testing would also be relatively easier, as higher priority tests can be prioritized and executed ahead of lower priority tests. Critical defects would also be identified early in the testing phase since the tests are prioritized, and as the test execution progresses, minor defects are identified. This would ensure that the critical defect leakage is minimal. Hence, the cost of fixing defects is also lower and leads to an increase in the confidence level in the quality of the application.

***Other benefits:***

Creation of reusable artifacts either based on the most recent testing assignment or as a separate activity in itself is also bound to be highly effective, as critical scenarios, test cases and effective test data can be identified with minimal effort, thus, forming an effective regression suite.

However, achieving this requires commitment from the management to retain resources. Some of the challenges involved in building or maintaining domain based testers are as follows:

- The whole process of building domain expertise among resources by an organization can only happen over a period of time. Especially where software testing activities are outsourced, it requires commitment from the vendor's management to retain resources for projects in a specific domain and this could lead to higher resource holding costs.

- Building domain expertise requires focus and effort from the employee as this learning will not happen completely within the scope of a project. The tester will have to consider reading additional material or better still completing domain certifications.

- From the customer perspective, if the maturity of an industry is low, the testers are confined to one domain and they cannot learn best practices from other domains. However, this can be easily overcome if the organization documents and implements the best practices.

Case Study 1: In one of the projects that we have executed for a life insurance client, the legacy application was developed in-house

and was designed to give complete group life insurance functionality with more than 180 CICS screens and 520+ batch jobs. Adding to the complexity were client-related customizations and limited documentation.

As an independent testing team, we had limited time to familiarize with the application and needed to be productive from week 1. In this situation, it was a big advantage that the team had domain knowledge. Based on this, we created a knowledge acquisition framework for the application.

We then assigned priority to knowledge acquisition based on the immmediate project needs and assigned owners for exploration and documentation of the knowledge gained. This was then used to cross-train other testing team members. This way we were able to ramp up resources in a matter of two weeks. As we included sufficient quality checks, there was zero defect leakage in a project with a testing effort of more than 72 person months and four releases. At the end of a couple of projects, the application documentation was also helpful in subsequent ramp-ups.

Case Study 2: For one of our general insurance clients, we have taken over production support testing from the existing industry package vendor for a system which provides functionality for creation of insurance products, new business acquisition and policy administration. Testing services were provided for releases, emergency builds and enhancements for mainframe and the Java-based application. The challenge here was the uneven flow of work for the testing team. The team size cannot easily fluctuate with the flow of work, as the development team size was close to 9 times bigger than the testing team. Moreover, a seemingly minor change may require a huge amount of regression testing. We created a scenario based estimation model based on domain knowledge for different lines of business. This was helpful in projecting and controlling the workflow for the testing team, and it was easier for the testing team to negotiate the flow of work. If need be, we were in a position to share the high level scenarios to substantiate our arguments when planning for the build. This avoided last minute surprises.

The benefits of domain based testing outweigh the constraints generated by it. All the above factors would, undoubtedly, contribute to high quality applications, lower cost of quality and, equally important, better testing experience!!

## > biography

**Girish Valluri**
*is a Post Graduate in ComputerAapplications from Bangalore University with eight years of experience in software testing for financial services. Certified in testing and property and casualty insurance,he has been leading software testing projects in general insurance and life insurance for more than six years.*
*As a lead of the Insurance Testing Center of Excellence at Capgemini India, he is responsible for capability building, spearheading development of solutions and participating in proposals. These activities cover all insurance sub-domains( Life, General & Health) and key industry packages in the above areas.*
*As part of earlier roles, he has - apart from regular functional testing and automated testing - led teams and managed projects for user acceptance testing, production support testing, legacy modernization, test environment sanitization on Legacy and distributed platforms. Also, he has managed testing engagements from client's offices and offshore locations and was involved in all project phasesright from initiation through to closure for custom insurance applications and industry packages.*

# Economical turmoil a.k.a. the ecstasy of QA in the telecommunications field

*by Attila Fekete*



In certain cases being the pioneer of something might give you an advantage over competitors. In other cases, however, it is clearly a disadvantage. To a certain extent this is the case for the pioneers of the telecommunications industry. They started the business when no generic solutions were available on the market. Then target HW, SW and even proprietary programming languages were developed. Once you have products based on old technologies it is pretty difficult, time-consuming and expensive to make the shift from old to new technologies. The same applies for processes and methodologies. Partly due to this and other reasons, software quality assurance was and still is a problem area in the telecommunications industry.

### From paradise to real life

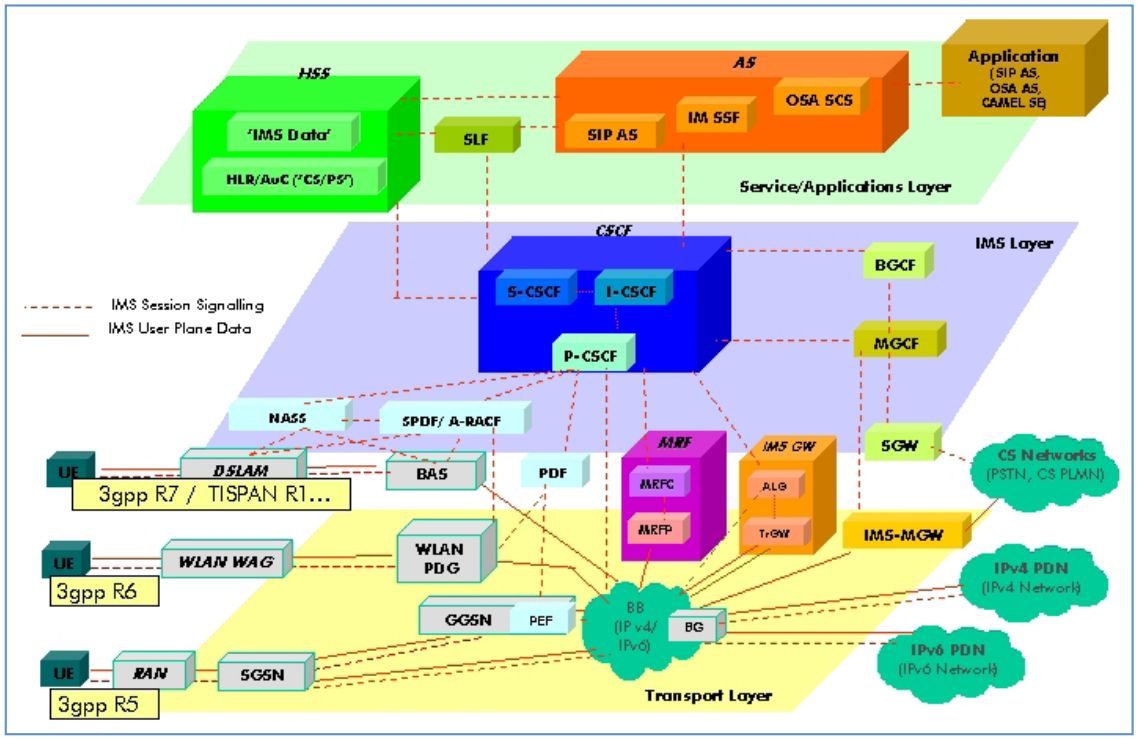
There was a spectacular rise in share prices starting from the mid-nineties up until 2000 when the first crisis hit the industry. Share prices were up to previously unknown heights and most players believed it would last forever. Now, if you have seemingly enormous amounts of money, you might lose focus on critical areas such as efficiency and innovation. Not driving in the fast lane of the highway of quality assurance anymore, but changing back to 4th or even 3rd gear. Quality assurance was not prevention oriented but rather a kind of final evaluation. Due to this wrong mindset quality assurance was rather expensive. Fortunately, crisis hit the industry and pushed the nitro button on our old 1969 Chevrolet Corvette.

### Proprietary system components

Due to historical reasons, as mentioned before, the majority of system components were developed in house. As a consequence of this, heavy all-around testing was needed. Since the HW itself was often proprietary, the simulated environments, development environments and debugging tools also needed to be proprietary. However, due to the telecommunication crisis around 2000 and the current economical turmoil, this has changed a lot. Since the early 2000's more and more third-party components are integrated into telecommunication systems. This way costs, lead-time and required testing effort have decreased significantly.

### Complexity

Telecommunication systems were and still are too complex. And as many of you working in the QA field know, the costs of assuring an acceptable quality level increases exponentially with the complexity of the system. It is hard to argue that the technology drivers of the telecommunication industry have had their heads in the clouds. This is neither good nor bad. Fact is that the business world and the technology world are sometimes too far away from each other to create a healthy balance. In many cases the real value of a certain feature was not in balance with the costs of development, software testing and maintenance. And to go even a step further, in certain cases even the real busi-



ness case was missing too. A pretty good example for the lack of balance is the well known 99.999% and 99.9999% availability that today's telecom grade systems offer. It is questionable that an everyday user would require this level of availability 24 hours a day, 365 days a year. In most cases users accept lower quality of service for significantly less money or even for free. There is a clearly visible trend of users turning to alternative communication solutions like Skype when communicating just for fun. Of course, when it is about life or expensive assets, that is a different story. So I think experience has shown that the wrong approach was chosen. In the battle between an all-in-one solution and separate systems for different use cases / demands, the latter came out as the winner. Often, less is more, and that is true for telecommunications too.

However, there is one more reason why it is challenging to test telecommunication systems. If you look at the figure, you will see that it tries to outline the main building blocks of a telecommunication system and you will quickly realize that a complete system is built up from many entities. For each and every feature several of these entities are involved at the same time. Due to this network, entities must be integrated as a whole system after node level testing. And if we go even a step further, telecommunication networks usually are not homogeneous, but network components are purchased from different vendors. So on top of this, there are interoperability issues too.

### *Standardization and competition*

As if that isn't enough, due to tough competition, certain features might be based on non-finalized standards. Sometimes even proprietary solutions are developed before the standard is available. So different vendors might have different competing proprietary implementations. This can make integration activities extremely difficult in certain cases.

### *Load testing*

Load testing gives us QA engineers another challenge. Imagine telecom servers which are able to serve several hundred thousands of users in a region of a country. Now you can imagine how challenging load and stress testing could be. And if you put not only quality but lowering the costs into the equation, then it is even more challenging.

### *In-house-developed vs COTS tools*

In certain cases you cannot chose from generic Commercial Off-The-Shelf (COTS) solutions to test your system, but need to either customize one, or even sometimes develop your own testing system. However, it would be silly to forget that sometimes it pays off to develop your own system in-house. Bear in mind that telecommunication companies have tens of thousands of employees, and license fees are often too high. So for telecommunication companies it is sometimes worth it to develop a test tool by themselves since they have the resources to develop it relatively quickly, and, - even more important, - they have their own critical user base . These in-house-developed products are nowadays based on standard languages and frameworks: J-Unit, C-Unit, C++, Java, Python, TTCN... Of course frameworks are customized with additional libraries to fulfill the needs of products to be tested.

### *Diversity of SW and HW to test*

What also comes into the already complex mix is the diversity of the components to test:

- Distributed systems
- Blade systems
- Proprietary OSs
- Web Apps
- Embedded systems
- C++, Java, Erlang, etc... applications
- Databases
- Radio and other transmission technologies

As you can see from this list, telecommunication systems are all-in-one systems from the quality assurance point of view. A complete telecommunication system consists of web apps and regular applications, critical and non-critical components, OSs, switching nodes, databases and a lot more. Anything you as a quality assurance professional can imagine.

### *Debugging*

"Praise a fair day at night". This could have been said by a QA engineer working in the telecommunication field. These systems forming the core of a telephony network are often very sensitive to timing issues. Even a few printouts or turned-on traces could change the system's behavior and hide the fault completely. And even without extra traces it is often pretty difficult to reproduce the same faulty behavior. So often it is time-consuming and difficult to do the debugging. You need to be really smart and innovative.

### *Back to the roots*

But let's get back to the basic idea in the title. Everything changes and so does the telecom industry. Sometimes bad things could turn to be good ones. In the case of QA in the telecom industry, I think today's economical turmoil is a pretty good thing. It made us much more innovative and much more effective. We have started to get rid of our old bad habits and become real leaders in the IT QA field. We rely more on testing in simulated environments, automate more and recognize software testing/quality assurance as a profession. So I think the economical turmoil gave us the long-awaited boost. And of course the challenge of better quality for less money and with significantly shorter lead-times. That is what the economical turmoil brought us. Since the barriers are set, the only flexibility we have is through innovation and our creativity. It might sound scary, but I really like it. So fasten your seat belt and enjoy your flight.

## > biography

**Attila Fekete**

*After graduating in Information Sciences in 1998 Attila Fekete started to work in the telecommunication industry. Since then he has worked as Troubleshooter, Test Automation Engineer and Test Analyst. He has honed his skills in every area of software testing starting from functional to non-functional, from manual to automated and from scripted to exploratory testing.*

*e-mail: fekete.attila@ymail.com*


# Testing in the pharmaceutical contract research organization domain

*by Julie Lacroix*

## Domain constraints

**Performing software testing in a Contract Research Organization (CRO)[1] changes your testing vision for the rest of your life.** The pharmaceutical industry is heavily regulated, and software testing used in support of clinical research is not an exception. First, the vocabulary used in the pharmaceutical world is quite different; test plans and test reports are known as validation protocols and validation reports; user acceptance testing is known as validation testing, etc... Secondly, one of the worst nightmares of IT professionals working in CROs is THE DOCUMENTATION. Everything needs to be documented to demonstrate what actions were really performed. One rule in the pharmaceutical world is: "If it's not documented, it doesn't exist". So, you can imagine how much paper, printouts, and test cases need to be produced to prove the existence of testing activities on software used in support of clinical studies. In addition, every activity performed as part of testing needs to be clearly explained and documented in Standard Operating Procedures (SOPs). These procedures are regularly inspected and challenged by clients and agencies during audits and inspections.

Regulatory agencies' rules and guidelines also add constraints to software testing. The most known regulatory agency is the FDA[2] (USA). CRO clients, however, come from around the world; therefore testing procedures must be compliant with many regulatory agencies. One of the most famous regulations in the pharmaceutical domain is the CFR21 Part 11[3]. This regulation covers how software and companies must deal with electronic records and electronic signatures for clinical study data. Security of data, training of staff responsible for developing, maintaining or using the software, audit trail properties, and components of electronic signatures are explained as part of the regulation. Additionally, clinical study data (in paper or electronic form) needs to be readily available during the retention period required, i.e., 25 years. I'll let you imagine the backward compatibility challenges, and associated testing issues, where electronic records are concerned.

## Domain expectations

In order to help the industry to comply with these standards, the FDA published guidance on software validation in 2002[4]. This guidance explains the FDA's point of view and their expectations regarding software validation. In this guidance, principles of software validation are listed and explained. Table 1 presents the 10 principles and a short description of each. The European Commission Health and Consumers Directorate-General[5] also have created a guideline regarding computerized systems (Volume 4 Annex 11[6]). This guideline was issued in 1997 and revised in January 2011. The guideline lists 17 principles related to computerized systems. From these principles, the fourth one concerns software validation and is subdivided into 8 key elements. Table 2 presents these key elements from the European Commission and a short description of each. When both guidelines are compared, the principles and expectations are mostly equivalents. However, the new edition of the European Commission guideline adds more details. This may result in new approaches for future audits performed by European regulatory agencies.

## Algorithme Pharma Approach

Algorithme Pharma[7], a CRO located in Montreal, Canada, appointed a software validation team (known as the testing team to the IT world) to deal with these industry expectations for software testing. Software used by Algorithme staff ranges from in-house developed to well-established vendor-supplied, from desktop to web-based applications, and from clinical to biostatistics applications. This rich environment requires adoption of a strong and efficient testing strategy to ensure that all these applications are compliant with regulations and behave as expected. The approach adopted by Algorithme Pharma is requirements-based

1 *http://en.wikipedia.org/wiki/Contract_research_organization*

2 *http://www.fda.gov/*

3 *http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=11&showFR=1*

4 *http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM085371.pdf*

5 *http://ec.europa.eu/health/documents/eudralex/index_en.htm*

6 *http://ec.europa.eu/health/files/eudralex/vol-4/annex11_01-2011_en.pdf*

7 *http://www.algopharm.com/*

Table 1. Principles of Software Validation FDA

| Principles of Software Validation | Description |
|---|---|
| Requirements | Documented software requirements specification provides a baseline for both validation and verification. The software validation process cannot be completed without an established software requirements specification. |
| Defect Prevention | Software quality assurance needs to focus on preventing the introduction of defects into the software development process and not on trying to "test quality into" the software code after it is written. |
| Time and Effort | To build a case that the software is validated requires time and effort. Preparation for software validation should begin early. The final conclusion that the software is validated should be based on evidence collected from planned efforts conducted throughout the software lifecycle. |
| Software Life Cycle | Software life cycle contains specific verification and validation tasks that are appropriate for the intended use of the software. The guidance does not recommend any particular life cycle models – only that they should be selected and used for a software development project. |
| Plans | The software validation process is defined and controlled through the use of a plan. The validation plan defines "what" is to be accomplished through the software validation. |
| Procedures | Procedures establish "how" to conduct the software validation effort. The procedures should identify the specific actions or sequence of actions that must be taken to complete individual validation activities, tasks and work items. |
| Software Validation After a Change | When any change is made to the software, the validation status of the software needs to be re-established. Whenever software is changed, a validation analysis should be conducted not just for validation of the individual change, but also to determine the extent and impact of that change on the entire software system. |
| Validation Coverage | Based on the software's complexity and safety risk – not on firm size or resource constraints. The selection of validation activities, tasks and work items should be commensurate with the complexity of the software design and risk associated with the use of the software for the specified intended use. |
| Independence of Review | Validation activities should be conducted using the basic quality assurance precept of 2independence of review". When possible, an independent evaluation is always better, especially for higher risk applications. |
| Flexibility and Responsibility | Specific implementation of these validation principles may be quite different from one application to another. Software validation process should be commensurate with the safety risk associated with the system, device, or process. |

Bibliography

FDA. (2002). General Principles of Software Validation; Final Guidance for Industry and FDA staff.

Table 2. European Principles of Software Validation

| Principles of Software Validation | Description |
|---|---|
| 4.1 Life Cycle | Validation documentation and reports should cover the relevant steps of the life cycle. |
| 4.2 Change Control | Validation documentation should include change control records and reports on any deviations observed during the validation process. |
| 4.3 Inventory | An up to date listing of all relevant systems and their functionality should be available. |
| 4.4 User Requirements Specifications (URS) | URS should describe the required functions of the computerized system and be based on documented risk assessment and impact. |
| 4.5 Quality Management | Ensure that the system has been developed in accordance with an appropriate quality management system. |
| 4.6 Customized Computerized Systems | Process is in place that ensures the formal assessment and reporting of quality and performance measures for all life-cycle stages of the system |
| 4.7 Evidence | Evidence of appropriate test methods and test scenarios should be demonstrated. |
| 4.8 Migration | If data are transferred to another data format or system, validation should include checks that data are not altered in value and/or meaning during the migration. |

Bibliography

European Commission. (2010). Volume 4: Good Manufacturing Practice Medicinal Products for Human and Veterinary Use, Annex 11: Computerized Systems.

testing[8] completed with a risk-based testing technique[9]. Software used as part of clinical study is assessed against the compliance level of the application for 21 CFR part 11 and the risk related to this application.

The risk is evaluated on five factors:

- criticality on human life
- regulatory experience
- vulnerability if down
- industry distribution
- type of data (electronic, paper, mix of both) processing into the software

Once these factors are identified and evaluated, a risk level is determined. The risk analysis determines the extent of testing needed to satisfy both regulatory and internal quality standards. Criteria used to determine the testing level cover all Software Development Life Cycle (SDLC) phases, activities and good practices: requirement definition, formal design activities, peer reviews, known bugs, regression testing, acceptance testing etc... The goal of this approach is to maximize testing activities to reduce risk to an acceptable level (risk zero is the ultimate goal). The risk-based approach helps to identify the applications or parts of given software the testing team needs to devote more attention to and test more deeply. When an acceptable balance between risk and testing is reached, for a particular piece of software, this equilibrium needs to be preserved during subsequent software updates. For example strong change control procedures must be followed and regression testing must be performed.

The testing team at Algorithme Pharma has a double role. The primary goal of the team is to perform validation testing on applications used as part of clinical studies. The goal of the validation is to confirm the "fitness for use" of the application for its intended function. The second role is to perform verification testing for software developed in-house. Both roles require different types of expertise and testing techniques. Therefore different professional backgrounds are necessary in the testing team. To ensure testing expertise, Algorithme Pharma adopted and supports the ISTQB®[10] certification. This certification scheme was chosen over the other ones available due to its worldwide recognition and for being accepted as an industry standard. The background to assess the "fit for purpose" aspect is quite different to the pure software testing background. The ideal background for the "fitness for use" is a subject matter expert working directly in operations. However, such power users cannot work in the operations and test software on a regular basis. The compromise adopted by Algorithme Pharma was to build a testing team to design and execute verification tests (system testing) and to design validation tests (acceptance testing), while power users are responsible for the execution of the validation tests. However, testing staff needs to acquire a deep knowledge of the operations in order to design realistic and appropriate test cases. This knowledge is gained by assisting in departmental training and by observing operational tasks performed by the power users on their day to day operations. Another challenge for the "fitness for use" goal resides in the variety of functions existing in the company: clinical operations, bioanalytical laboratory, biostatistics and quality assurance. Different software solutions exist for these departments, therefore the testing team needs to understand each department's operation rules in order to design good test cases. To reduce the amount of scientific knowledge required by each tester, the testing team staff is assigned to projects based their domain knowledge to keep the testers in their area of expertise as much as possible.

The testing team is also involved in the requirements analysis phase for each project. This allows the testing team to understand the requests for change directly from the users, and to review final requirements based on their understanding. Therefore requirement static testing can be effectively performed. It also reduces the time needed for test case design since the testing team already knows the scope and content of the latest software iteration. The iterative SDLC model and the early involvement of the testing team allow the design, code and test case design to be performed in parallel (V-model)[11]. Testing techniques performed on all software versions include some exploratory testing and extensive scripted verification tests at system level and very detailed scripted test cases on user acceptance level in order to achieve 100% requirements-based testing and to address risks identified during initial analysis.

The software development model adopted by Algorithme Pharma is the "iterative time box"[12]. This model was preferred for its short return on investment (ROI) time, change management and to satisfy evolving business rules. For the users, this development model permits new functionalities to be put into production rapidly and to improve their efficiency. Two user acceptance testing techniques are performed by the power users: a formal execution of designed tests cases (requirement-based testing), and a more informal user exploratory session (documented in what's called "User Acceptance Report"). The later technique is performed using a copy of live data. Therefore, a user can choose a real life complex project and confirm that the new version fixes and/or improves the situation. This additional testing activity raises the confidence level of the users regarding the new software iteration by acting like early training and facilitates the implementation of the iteration into operation.

## So, what's the difference?

Algorithme Pharma established an efficient and compliant approach for software testing. By combining the requirements-based approach, considered as one of the most rigorous testing techniques, with the risk-based approach, software used by Algorithme Pharma for clinical study meets regulatory agencies' expectations and internal standards. The Algorithme Pharma testing approach has successfully passed audits by both our customers and regulatory agencies.

What is the difference between software testing in a pure IT domain and in the pharmaceutical domain? There is not much difference regarding testing techniques, tools, concepts, etc... One of the differences resides mainly in the extension of residual anomalies in software, i.e., defects that escape testing. Let's say an anomaly is introduced in a calculation in biostatistics software. A drug formulation meets the acceptance criteria of regulatory

*8 An approach to testing in which test cases are designed based on test objectives and test conditions derived from requirements. ISTQB Glossary of Testing Terms, version 2.1, April 1st, 2010.*

*9 http://en.wikipedia.org/wiki/Risk-based_testing*

*10 http://istqb.org/display/ISTQB/Home*

*11 A framework to describe the software development lifecycle activities from requirements specification to maintenance. The V-model illustrates how testing activities can be integrated into each phase of the software development lifecycle. ISTQB Glossary of Testing Terms, version 2.1, April 1st, 2010.*

*12 Rapid Development, Steve McConnell, Microsoft Press, 1996*

agencies following a clinical study. The drug formulation could be approved and will be distributed in pharmacies and hospitals. In the worst case scenario, the drug formulation may cause the death of patients. Because human life can be impacted somewhere, software testing in the CRO domain must be taken very seriously. The domain requires a very formal process to document the testing activities. The documentation aspect is really the main difference. The documentation must be able to reproduce, step-by-step, the testing phase performed before the deployment of software in the operations environment. Testing is not enough in the domain, testing must be accompanied by the appropriate documentation. In conclusion, since a picture is worth a thousand words, this cartoon represents the documentation reality in the pharmaceutical domain...


## > biography

**Julie Lacroix**
*has over 12 years of experience at Algorithme Pharma including 6 years as a Software Validation Manager. She is also a board member of the Canadian Software Testing Board (CSTB) as the Vice-President Examinations. Julie's competencies are team management, system and user acceptance testing along with regulations and compliance. Her previous experience in the Bioanalytical Laboratory of Algorithme Pharma helps her to better understand user expectations for software. Julie is ISTQB certified at the Foundation Level and at the Advanced Level as Test Manager and as Test Analyst. http://ca.linkedin.com/in/julielacroix3258*

HANDBUCH

# TESTEN
## IN DER FINANZWELT

*Das Qualitätsmanagement und die Software-Qualitätssicherung nehmen in Projekten der Finanzwelt einen sehr hohen Stellenwert ein, insbesondere vor dem Hintergrund der Komplexität der Produkte und Märkte, der regulatorischen Anforderungen, sowie daraus resultierender anspruchsvoller, vernetzter Prozesse und Systeme. Das vorliegende QS-Handbuch zum Testen in der Finanzwelt soll*

- *Testmanagern, Testanalysten und Testern sowie*
- *Projektmanagern, Qualitätsmanagern und IT-Managern*

*einen grundlegenden Einblick in die Software-Qualitätssicherung (Methoden & Verfahren) sowie entsprechende Literaturverweise bieten aber auch eine „Anleithilfe" für die konkrete Umsetzung in der Finanzwelt sein. Dabei ist es unabhängig davon, ob der Leser aus dem Fachbereich oder aus der IT-Abteilung stammt. Dies geschieht vor allem mit Praxisbezug in den Ausführungen, der auf jahrelangen Erfahrungen des Autorenteams in der Finanzbranche beruht. Mit dem QSHandbuch sollen insbesondere folgende Ziele erreicht werden:*

1. *Sensibilisierung für den ganzheitlichen Software- Qualitätssicherungsansatz*
2. *Vermittlung der Grundlagen und Methoden des Testens sowie deren Quellen unter Würdigung der besonderen Anforderungen in Kreditinstituten im Rahmen des Selbststudiums*
3. *Bereitstellung von Vorbereitungsinformationen für das Training „Testing for Finance!"*
4. *Angebot der Wissensvertiefung anhand von Fallstudien*
5. *Einblick in spezielle Testverfahren und benachbarte Themen des Qualitätsmanagements*

*Herausgegeben von Norbert Bochynek und José M. Díaz Delgado*

*Die Autoren*
*Björn Lemke, Heiko Köppen, Jenny Siotka, Jobst Regul, Lisa Crispin, Lucia Garrido, Manu Cohen-Yashar, Mieke Gevers, Oliver Rupnow, Vipul Kocher*

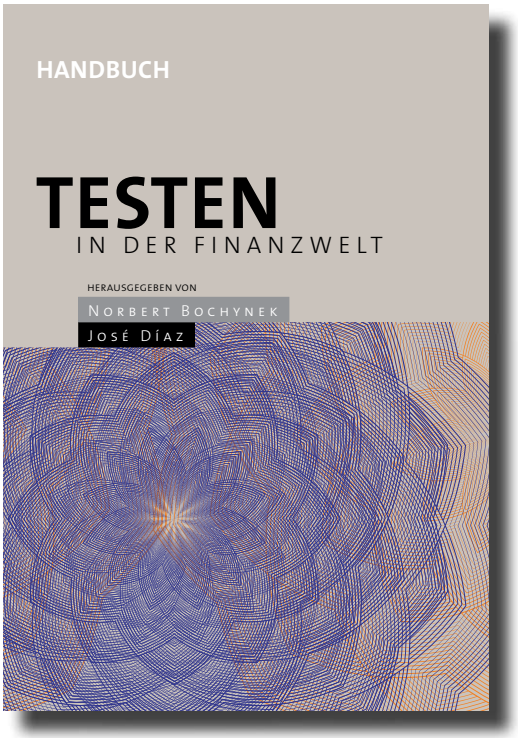
*Gebundene Ausgabe: 431 Seiten*

*ISBN 978-3-00-028082-5*

*1. Auflage 2010 (Größe: 24 x 16,5 x 2,3 cm)*

*48,00 € (inkl. Mwst.)*

*www.diazhilterscheid.de*




# Dairy of a user tester

*by Carsten Wium & Lucas Hansen*

*Dear diary*

Good news! Today my boss told me that he would be sending me out to test our new software solution. He told me how quality minded and thorough I am. He said that the guys at the IT department had been working for a good while on this piece of shiny new software, and he wanted me to take it for a trial run. Feeling like my dad had just given me the keys to his new car, I promptly cleared my early summer schedule for the "User Test". I also readied a speech for my wife about how we would have to wait another season before going to France.

This is going to be great!

*Dear diary*

I was at the IT department today for my first planning meeting. I must admit my first impressions were met in full. There were computer geeks all over the place, coffee machines to feed them all and not a customer in sight! When I got to the meeting room, I met with the Test Manager – a bright one with a nice tie and the good manners too. He kicked off the meeting by letting us all know how important we are, and I must admit, it felt nice. We were all from different branches and we all felt much appreciated for taking time out of our calendars to be here. Some had been testers before, while others like me were here for the first time.

Using several complicated terms, the test manager told us all of the plans for the test. I'm not really sure what they were, but the power points looked good. Tomorrow, everything is going to make much more sense.

*Dear diary*

Day three and we are almost done with the preparations for the test. Surprisingly enough, today wasn't at all the best of days, after the first day's confusion. The test manager showed up with a "business developer" who had a few moments to talk about the new software. It sounded good, but she had to run shortly after the presentation. Then the test manager started talking about another test which was making our test more difficult. Apparently someone was doing a system test. I'm not sure why that would be necessary since we are all here to do the testing, but in any event, that test was more important than the remaining introduction and we were sent home early.

*Dear diary*

It has been two weeks since we finished our planning and preparation and there are now three days to the beginning of the test! Here in our department, everybody is feeling tired of the old software and they are looking forward to my stories when they return from their holidays.

Anyway, at the end of the day, I got a mail saying that the test is postponed for two weeks. The new software is having some bugs fixed (which, incidentally my cat also had this spring) and the developers needs a bit more time.

*Dear diary*

Today is the first day of the test, and I learned a valuable lesson. Showing up early has its rewards. Apparently someone else had made a miscount of how many testers were coming, and the last four testers that came through the door were divided onto only two computers.

My first test case was a bit difficult to find. Our training with the test software had been much simpler than it really was. Apparently there was a bit of difference between how the developers sorted these "test cases" and how we practiced it during the training. When I finally found what I needed, I was good and ready to get to work like the thorough and quality minded tester that I supposedly was.

There was plenty of work cut out for all of us. I tested many of the work processes that go on in my branch. A few I've had much experience with and some I got to know a lot better. Today, I also found a defect in the approval of invoices. Apparently the system allowed for all users to approve invoices, regardless of their position, and I am not sure that my boss would like for the canteen to approve his orderings of new hardware. This was my first actual defect and it felt good to catch that one before the software was released.

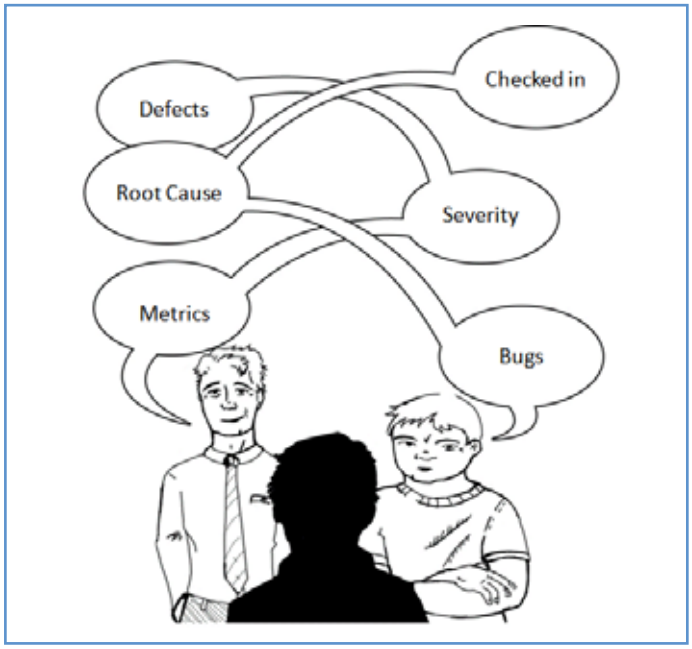
*Dear diary*

Day two is all about testing. The test manager comes by from time to time and asks how we are doing. I tell him that we are doing fine and ask whatever questions I may have. His answers sometimes leave me wondering what I had been asking in the first place, but seeing as he knows best, I'll try to figure out the higher meaning as I go along.

At the end of the day, the test manager returns to show us some graphs about how well we are doing. There is a green column which is good and a blue one which is taller – apparently he had hoped the two would be the same height. Then there is the red one – that's how many defects we had.

*Dear diary*

Today I was a bit late, but since we are doing our own scheduling, I guess that is ok. When I got to my computer, I find that a developer has entered the test. He is a short guy wearing a tee and jeans, drinking coffee at the same rate others breathe air and he doesn't look the least bit happy. In fact he looks rather distressed and is also, in fact, waiting for me. It turns out that the defect I found has caused him some concern and made him walk "all the way over here".




While he is talking fast, I find myself drifting away with his notion of "code that hasn't been released", enjoying the visual image of 1's and 0's escaping some sort of cyberspace jail. When the test manager interrupts, it turns out that my defect wasn't really a defect at all. Apparently, part of the software just wasn't done yet.

*Dear diary*

Today I left after the last day of the user test. Stuck in traffic I noticed a poster on a bus stop marketing the software I have been testing.

I imagine such things would excite me on most days, but today I feel a twist in my gut. At our last status meeting, the graphs the test manager showed had a fairly high red bar, which meant that I haven't been the only one to register defects. With his nice tie and pleasant attitude, he was thanking us for all the work we put into the test and noted that the developers would fix the defects and that they would still release on time.

I wasn't among those who got invited to the retest period where they would make sure all the known defects were fixed, and it leaves me wondering: Will I see some of these defects when they finally install the new software? And am I going to explain to my colleagues why the software isn't working as it should?

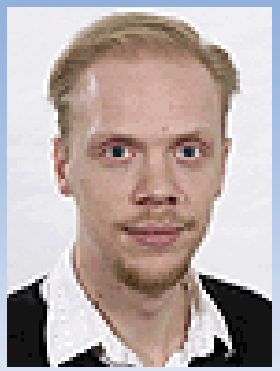
*Dear diary*

It has been three months now since the test. The first few days back in the branch were pretty hectic. Between my regular tasks and the familiar getting to hear all the stories from everyone's holiday, I was doing a mix of promotion of the new software and telling my stories from the test.

To my surprise, I return from lunch one day to find that everyone has received a mail with an updated work process with regard to invoice approvals - So my very first defect was never corrected! - or the developer maybe misunderstood me. I somehow feel that I should have followed up on this when I was doing the test. During these first three months, the software had several "upgrades", most of which brings to mind the discussions and questions that I and other testers were asking. Today the new software (almost) works as well as the marketing campaign promised – and I am proud to have been part of making it happen.

## > biography

**Carsten Wium**
*is a Program Test Manager with 24 years of working experience within software development projects. Carsten has been responsible for managing project programs with up to 70 projects, and has developed a series of standard processes, templates for documentation, usage of test tools as well as training within quality assurance & quality control.*
*Carsten Wium has managed complex behavior change projects to increase awareness regarding test.*
*Carsten has presented various test related topics at Software conferences in Denmark since 2006, and is currently employed by Danske Bank as Program Test Manager.*



**Lucas Hansen**
*is a Test Manager with 4 years of working experience within software development projects. Lucas Hansen is currently employed by Danske Bank as Test Manager.*


# The key principle of testing "Testing is context dependent"

*by Nisha Dhiraj Kale*

Today's test teams have to cope with the increasing complexity of the systems under test, while often schedules, budgets and team sizes have not been scaled accordingly. Not every part of the tests can be automated and, due to limitations in time and budget, only a small fraction of all use cases can be covered in manual testing. This article describes the domain knowledge necessary for a software tester and its advantages, particularly with regard to helping the testers manage the complexity issues and select and prioritize the tests according to the application under test.

## Software Testing Principles:

Software testing is an extremely creative and intellectually challenging task. When testing follows the following principles given in the ISTQB® Foundation Level syllabus, the creative element of test design and execution rivals any of the preceding software development steps.

### *Testing shows the presence of bugs*

Testing an application can only reveal that one or more defects exist in the application. However, testing alone cannot prove that the application is defect free. Therefore, it is important to design test cases which find as many defects as possible.

### *Exhaustive testing is impossible*

Unless the application under test (AUT) has a very simple logical structure and limited input, it is not possible to test all possible combinations of data and scenarios. For this reason, risk and priorities are used to concentrate on the most important aspects to test.

### *Early testing*

The sooner we start the testing activities, the better we can utilize the available time. As soon as the initial products, such the requirement or design documents are available, we can start testing. It is not uncommon for the testing phase to get squeezed at the end of the development lifecycle, i.e. when development has finished, so by starting testing early, we can prepare testing for each level of the development lifecycle.

Another important point about early testing is that when defects are found earlier in the lifecycle, they are much easier and cheaper to fix. It is much cheaper to change an incorrect requirement than having to change a functionality in a large system that is not working as requested or as designed!

### *Defect clustering*

During testing it may be observed that most of the reported defects are related to a small number of modules within a system that contain most of the defects found. This is the application of the Pareto Principle to software testing: Approximately 80% of the problems are found in 20% of the modules.

### *The pesticide paradox*

If you keep running the same set of tests over and over again, the chances are that no further new defects will be discovered by those test cases. Because as the system evolves, many of the previously reported defects will have been fixed and the old test cases do not apply anymore. Anytime a fault is fixed or a new functionality added, we need to do regression testing to make sure the new changed software has not broken any other part of the software. However, those regression test cases also need to change and reflect the changes made in the software, in order to be applicable and hopefully find new defects.

### *Testing is context dependent*

Different methodologies, techniques and types of testing are related to the type and nature of the application. For example, a software application in a medical device needs more testing than a games software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulations and possibly specific test design techniques. By the same token, a very popular website needs to go through rigorous performance testing as well as functionality testing to make sure the performance is not affected by the load on the servers.

### *Absence of errors fallacy*

Just because testing didn't find any defects in the software, it doesn't mean that the software is ready to be shipped. Were the executed tests really designed to catch most defects, or where they designed to see if the software matched the user requirements? There are many other factors to be considered before making the decision to ship the software.

In software engineering, domain knowledge is knowledge about

the environment in which the target system is to operate, and every application applies different methodologies, techniques and types of testing which are related to the type and nature of the application. Different software products have varying requirements, functions and purposes, so the same test cases and strategy cannot be applied across all applications. For example, a software application in a medical device needs more testing than games software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulators and possibly specific test design techniques. By the same token, a very popular website, needs to go through rigorous performance testing as well as functionality testing to make sure the performance is not affected by the load on the servers.

Testing as a job requires one to be proficient with technical knowledge and experience of the development life cycle, functional domain, and technology and tools used for testing. In addition, it requires individuals to demonstrate a high capability for logical analysis, deduction, reasoning, communication, and presentation skills.

Business domain knowledge is important as it ensures that the tester understands the perspective and needs of the users for which the software is being built. Good business domain knowledge ensures that testers are more likely to identify missing requirements and incomplete requirements or stories that might need extra details; they can identify relevant test requirements/scenarios and create effective test cases. Communication between end-users and software developers is often difficult. They must find a common language to communicate in. Developing enough shared vocabulary to communicate can often take a while.

When we test a particular product, we have two kinds of axes where Y axis denotes the technology and where the X axis denotes the domain. Domain is common for all technologies. Domain experts can write the business driven scenarios to execute the actual business, whereas the technology experts will be able to write scenarios on the technology, which makes their scope narrow.

## The advantages of domain knowledge:

1. Reduces the training time: A person can be productive quicker than a person that has zero domain/industry knowledge. So it adds value to project/product.
2. Good knowledge of the functional flow (workflow), the business processes and business rules: It will help to better understand the product requirements.
3. Good knowledge of UI features: It will help the person in improving the look & feel of the UI as well finding more bugs at an initial stage at the UI front-end.
4. Good knowledge of back-end processing: Knowing how effectively/efficiently the data/code is handled.
5. Domain knowledge is also important for defect triage: Knowing how the application will likely be used and how it is expected to perform, will tell QA whether a given defect is trivial, significant, or in fact critical.
6. Terminology: Reporting in the language of the business, resulting in good rapport with the business team(s).

Are you going to test the BFSI applications (Banking, Financial Services and Insurance) just for UI or functionality or security or load or stress? You should know what the user requirements in banking, working procedures, commerce background, exposure to brokerage etc. are, and you should test the application accordingly. Only then you can say that your testing is enough. This is where domain experts are needed.

Let's take example of my current project: I am currently working on a stock market application, where I need to know the basic concept and terminology used in stock markets. If I know the functional domain better, I can better write and execute more test cases and can effectively simulate the end user actions. This is a distinct advantage. As testers we need to apply our domain knowledge in each and every step of the software testing life cycle.

While testing any application you should think like an end-user. The user who is going to use your application may have a good understanding of the domain he is working in. You need to balance all these skills so that all product aspects will get addressed.

If you have the above skills, you will certainly have the advantage of domain and testing experience, and therefore have a better understanding of different issues and deliver the application better and faster.

## Case Study: Testing BFSI applications (Banking, Financial Services and Insurance)

Testing has always been challenging in the software development life cycle of a product. Testing financial products requires more domain knowledge, and testing in general is very critical in the life cycle of any software development activity. Compared to other domain products, financial products pose significantly more challenges due to their complexity in design, development and testing, impact on end users, and safety characteristics. This in turn means that the test engineers face multiple challenges while testing the financial software to deliver high quality to the customer.

Some of the challenges faced while testing in the financial domain are:

| Id | Challenge | Explanation |
|---|---|---|
| 1 | Domain and System Knowledge | Lack of domain and system knowledge will result in inadequacy of testing. Test engineers with testing knowledge will not only suffice testing the product as a whole but also must know the complete functionality, usage scenarios, workflows, environment, standards used, configurations, regulations, interoperability, domain tools etc. |
| 2 | Application Workflow | Testers must have good understanding of usage and workflow use cases of the product as used by the end users at there sites. Bugs raised on invalid use cases may be rejected which leads to waste of effort in finding and reporting the bugs. |
| 3 | Test data/ datasets | Lack of availability of standard test data/datasets during testing will lead to insufficient test coverage. Test engineers must have standard datasets classified based on various parameters stored at a shared central repository and make it available for testing to ensure adequate coverage of tests at various test levels. |
| 4 | Financial domain standards | Insufficient knowledge of domain standards will result in poor quality of testing. |
| 6 | Specialization tests i.e., Image quality, Performance, Reliability, interoperability, Concurrency, Hardware etc., | Financial products have to be tested for some specific specialized tests to meet the quality goals of the product. Test engineers must be experienced fully trained and possess special skills to perform these tests. |
| 7 | Process Compliance - ISO, CMMI | Financial devices are compliance to certain standards to certify that they are fit for intended use. Test engineers must be trained on these standards to perform their job well in qualifying the product during testing. |
| 8 | Exposure to end users/Application specialists and Site. | Lack of exposure to end user use cases, interactions with Application specialists and non-awareness of sites will result in insufficient knowledge of product and poor quality of tests. |
| 10 | Unstable releases | Developers check in software and test team is made responsible for building the software and verifying it. Developers did not do sufficient unit testing before checking in. The code checked in was also not reviewed in many cases. Test engineers were unable to proceed on testing as per their plan as they faced many issues, which should have typically been caught during reviews and unit testing. |

## Conclusion:

Domain knowledge plays an important role in software testing, as one of the software testing principles says "Testing is context driven". Testing is always done differently in different contexts. Domain expertise is important in software testing because the person who has domain knowledge can test the application better than others.


### > biography

**Nisha Dhiraj Kale**

*As an ISTQB® certified testing professional I am currently working as Associate Consultant at Capgemini Pune for their Client CLSA. I have 5 years of testing experience in software testing. During this time, I have worked on different projects ranging from client server to web-based applications. I have knowledge of the lending, capital market and banking domains. I am an engineering graduate and also hold a diploma in Business Management from Symbiosis Pune.*

testing experience

Díaz Hilterscheid

# Already Certified?

## Join the Alumni Scheme and keep your knowledge up to date

- 50 e-books on Testing and related IT
- Regular content updates
- Discounts on Learntesting products
- Latest version of certificated courses
- 12 months 24x7 access
- For Individuals and Business
- Entry criteria includes all historical ISTQB and ISEB Foundation, Advanced & Practitioner Certificate Holders

### Special Testing Experience readers discount offer to 31st January 2011

***Foundation Alumni ~~€30~~ €20***

*Introductory offer, plus VAT*

***Advanced Alumni ~~€60~~ €40***

*Introductory offer, plus VAT*

**Visit www.learntesting.com**

using Promotion Code TEX001 and send a copy of your

certificate to helpdesk@learntesting.com

**Sign-up all your certified testers**

For details of our Corporate Alumni Scheme contact sales@learntesting.com

**The Learntesting Alumni Scheme**

'Supporting professionalisation of the Software Testing Industry'


© Katrin Schülke


# Testing and testers in the automotive sector

*by Fabrice Ravel & Dirk Schwarz*

This article deals with some of the particular aspects of testing in the automotive domain. By highlighting some specific aspects of the development and test process, the current automotive standards, the nature of test objects and the required skills and qualifications of the personnel, the authors show the profile of the "automotive tester" and demonstrate the challenge of test outsourcing in the automotive business.

## Test process

Considering the general development and test processes, one can say that most of the development activities in the automotive domain (and therefore also the testing) follow an iterative approach based on the V-model. Due to the nature of the product (mechatronical/embedded system with strong requirements for non-functional features), there are two main particularities impacting the right (testing) side of the V-model [1]:

1. Three development branches (mechanics, electronics and software) with high levels of interdependency.
2. Important and extensive calibration of control units at the end of the integration (i.e. the upper right side of the V).

The first aspect often results in difficulties for the test manager and test designer in finding the right test case abstraction level and test environment (e.g. test with a „software in test bed", test on breadboard with unit control stimulation, Hardware-in-the-Loop simulation), even if they are only in charge of testing the software. This implies a basic understanding of the system and its software, electronics and mechanics interactions, as well as skills in using specific test tools (e.g. CAN-data-bus measurement tools, HiL simulation, etc.).

The second aspect mentioned above refers to the high quantity of control unit parameters contained in the control unit software (up to several thousand parameters in complex control units). These must be set in order to define the behavior of the control systems and, as a result, fulfill both functional and non-functional system requirements. As an example, the dynamically changing map of the transmission control unit software contains all points of gear shifting combinations, depending on the gas pedal position and the vehicle velocity. The values in this map have to be defined optimally in order to achieve the target vehicle fuel consumption, performance and comfort requirements. Though the control unit calibration is executed by specialists for control systems and calibration, this activity has an important effect on the testing organization: Some requirements and features can only be validated when the end calibration values are set. On the other hand, it may not be possible to complete the calibration if the software functions do not work correctly and reliably. Therefore the jobs of the calibration specialists and testers need to be carefully defined, including a clear scoping of the test cases for the different integration levels.

Furthermore, the test of software-based systems in the automotive industry is impacted, among others, by a number of system features, including real-time requirements, distributed implementation of functionality, high quantity of physical input signals and applicable safety requirements. The test of these features (and in particular of the safety requirements) requires a proper and mature test process and the use of state-of-the-art methods and standards.

As soon as you are put in charge of testing safety relevant systems (e.g. airbags, brake systems, some functionalities of the power train like clutch automation), you will not get around the "Functional Safety Standard for Road Vehicles" ISO 26262. ISO 26262 is currently only available as a "Draft International Standard", but is practically already set as a reference. This standard is an adaption of the general Functional Safety Standard IEC 61508 to meet the needs of automotive electric/electronic system development. It includes precise advice concerning the test process and the test design techniques to be used, depending on the safety level (ASIL) of the system under test.

## Test design

In order to test automotive embedded software efficiently, many particular system features (some of them listed above) have to be taken into account.

It is typical for an embedded system to have a history-based or state-based behavior. Therefore the use of state-based test design techniques is a prerequisite (i.e. state-of-the-art) for a meaningful testing of automotive control unit software.

While choosing representative values for the test cases (e.g. with equivalence partitioning and boundary analysis test design techniques), one should keep in mind that a lot of input and output software data comes from or is converted into real signals with the help of the control unit electrical signal conditioning. However, some theoretical boundary values for software parameters cannot be represented in meaningful test cases, because they cannot be reached in reality.

The real-time requirements also have to be taken into account by planning and defining test cases. First of all you have to clarify which requirements are affected by real-time aspects and then you can choose the convenient test design techniques and the test environment. For this reason, the test designer, even at the software test level, needs to understand the basics of the control unit set-up and of the physical behavior of the system in order to define an efficient and effective test strategy [2].

## Outsourcing

Because of the rising share of software in the automotive industry and the growing complexity of the systems, it can be advantageous for a company at a certain point to outsource part of the test effort. Reasons for outsourcing some of the software tests may be, for instance, cost control, available expertise, independent testing, risk reduction or work load balancing [3].

To meet the challenge of minimizing the risks and emphasize the benefit for both company and vendor, they have to stipulate clearly and precisely the nature, scale and duration of the outsourced services. This includes, among other things, agreements on testability issues (e.g. background information for the tester, support, configuration and change management), management issues (e.g. who has final authority on testing priorities, deliverables, approvals) and other standard contractual issues (e.g. assignment of rights and responsibilities, confidentiality agreements, liability for the quality of the work).

Moreover, the test service supplier usually has to fulfil the following expectations to win a contract for outsourced test activities from an automotive customer:

- An adapted test infrastructure (e.g. HiL simulators coupled to a test management and test
- automation tool chain).
- Sufficient manpower reserve, which allows a flexible increase of the test work load without generating delays. This can help to cope with unexpected quality problems during development and is particularly important in a branch where the deadline "Start of Production" cannot be postponed without inducing huge costs.
- A high internal test process and test practice level, ideally confirmed by a corresponding CMMI or A-SPICE maturity level.
- A well-skilled test team with automotive specific background.

## Automotive tester

In addition to the usual general test expertise, the automotive software tester should have further specific competences, as introduced previously:

- A competent understanding of embedded and control systems in general, as well as a basic knowledge about the particular application field of the control unit under test. This is mandatory to be able to communicate efficiently with the requirements engineers and system designers, and to understand properly the software functions and derive meaningful test cases.
- A good knowledge of the specific automotive measurement and test tooling like, for example, CAN data logger, CAN-data-bus measurement tools, Hardware-in-the-Loop simulation chain, and software calibration tools.
- Knowledge of applicable automotive standards (e.g. Safety Standard ISO 26262, software coding rules MISRA-C) and best practices.

## Conclusion

Due to the increasing use of software in processes and products, software has a significant impact on product quality and company productivity. This is particularly true in the automotive industry. In order to meet the challenges and requirements of automotive software-based systems, an effective and efficient test process and test strategy must be applied. This can only be achieved by taking into account the specific aspects of the product and the sector. On the one hand, you have to master the state-of-the-art for the test of embedded systems. On the other hand, you must consider the trade-specific standards (e.g. MISRA-C, ISO 26262 ...), the specific organization and the development process which, in turn, influence the test methodology and the test technology used.

Automotive software testers are expected to have a high degree of competency and knowledge in, for example, general test subjects, the use of specific test tools and automotive and embedded systems. These are some reasons which make the job of an automotive software tester especially challenging and fascinating!

## References


1. „Embedded Systems qualitätsorientierte Entwicklung", Klaus Bender, Springer, 2005
2. „Systematisches Auswahlverfahren der Testmethodik zur Steuergeräte-Validierung", Jens Riese and Fabrice Ravel, Conference „Simulation und Test für die Automobilelektronik", Berlin, June 2010
3. „How to build and maintain a successful outsourced, offshored testing partnership", Dr. Mike Bartley, Testing experiences, March 2009

## > biography

**Fabrice Ravel**
*is a team manager at IAV GmbH, in Berlin, Germany. He is in charge of the topics requirements management and system integration for automotive transmission and hybrid systems in the business area Powertrain Mechatronics. He graduated as an engineer from the Technical University of Compiègne, France, and from the Technical University of Braunschweig, Germany, with a specialization in vehicle technology. After his graduation, he has worked on many automotive software and system test projects on different jobs, from test execution up to test management, and he received an ISTQB® full advanced tester certification. Currently he is mainly involved in development and test process consulting tasks.*

**Dirk Schwarz**
*is technical consultant for test methodology and test tools at IAV GmbH, in Berlin, Germany. He received his Dipl.-Ing. (FH) in mechatronics at the Friedberg University of Applied Sciences and his master degree in mechatronics from the St. Gallen Interstate University of Applied Sciences and Technology, Switzerland. He is an ISTQB®-certified tester and has several years of experience in software development and test. His personal interests are focused on test process improvement, especially methodological approaches.*

# IREB - Certified Professional for Requirements Engineering - Foundation Level

Díaz Hilterscheid

**Description**

The training is aimed at personnel mainly involved in the tasks of software requirements engineering. The tutorial is designed to transfer the knowledge needed to pass the examination to become an IREB CPRE-FL.

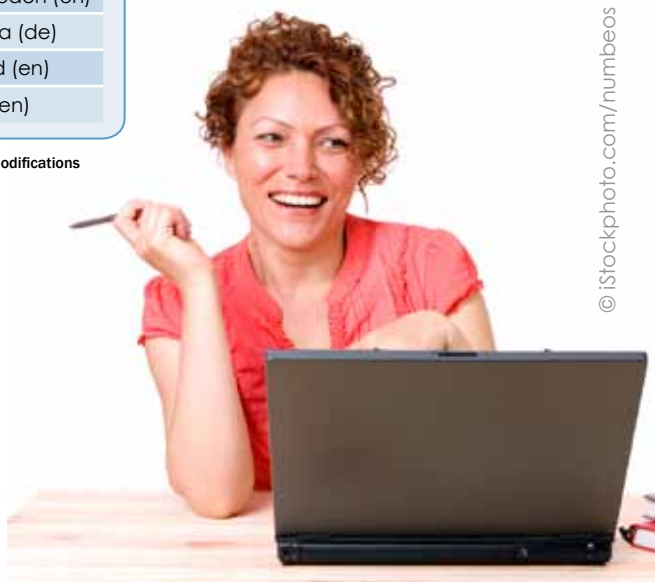
After earning a CPRE-FL certificate, a certificate holder can assess whether a given situation calls for requirements engineering. He understands the fundamental characteristics of the discipline and the interplay of methodological approaches, e.g. interview techniques, description tools or forms of documentation.

More information regarding the required knowledge can be found in the IREB Syllabus, which can be downloaded from the IREB web site: http://www.certified-re.com

http://training.diazhilterscheid.com

| Dates* | 3 days |
|---|---|
| 06.04.11–08.04.11 | Mödling/Austria (de) |
| 14.06.11–16.06.11 | Berlin (de) |
| 13.07.11–15.07.11 | Berlin (de) |
| 31.08.11–02.09.11 | Mödling/Austria (de) |
| 14.09.11–16.09.11 | Stockholm/Sweden (en) |
| 05.10.11–07.10.11 | Mödling/Austria (de) |
| 26.10.11–28.10.11 | Helsinki/Finland (en) |
| 23.11.11–25.11.11 | Oslo/Norway (en) |

*subject to modifications



# Testing = Tasting

Column

*by Thomas Hijl*

Today I gave a presentation at the office of one of our customers to a group of six people. One of them was a test manager working within their organization. At certain moments during the presentation we, the test manager and myself, got into the details of testing, sharing thoughts. I noticed by the look on their faces, that the others sometimes didn't have a clue what we were talking about. So, I explained the details to them in a more basic way.

The above situation made me realize again that it is always important to make the customer fully understand what we are doing as testers. Sometimes, we might get too far away from the basics and talk too much about sophisticated test techniques using jargon (difficult professional language).

A few weeks ago, for example, I was in Spain with my wife and little Julian. My boy caught "the flu" on the way there by plane. With a temperature of 40 degrees and rising, we rushed off to the local doctor. He explained very well what was happening and what he would prescribe. He also explained what side effects could occur etc. etc.. We were very pleased.

I remember the time when we went to the doctor and he didn't say much, had a look in your ear, looked into your eyes and throat, listened to your chest, sat down, scribbled something illegible on a piece of paper, and off you went with the paper to the pharmacy. Although this example may not directly link to testing, it certainly proves that there have been changes in the medical domain in this respect. They want to let people "taste" what they are doing to make you comfortable and to trust them in the work they provided.

Back to the beginning of my column,.... I think we can learn something here as testers.

### *Testing = Tasting*

Whatever the domain we work in, automotive, ICT, medical or industry, we need to let our customers "taste" what we are serving them. In the automotive industry this is already common practice, when people first make a test-drive before they buy a car. In my opinion we can give customers in the ICT and software business in many projects a better understanding of our work. Stick to basics, just like we would do at the local fruit & veg market, i.e. tasting a grape before buying the whole bunch. It's a very basic principle that will work for them.

Of course, it will be difficult sometimes to explain complex matters that we might work on. However, tasting will make your life easier.

### *The basics*

What is tasting about:

- Using more senses
- It's a first example of what's coming next, i.e. the rest
- It's physical
- It doesn't take long to decide

As far as my experience goes, I would say we should always stick to those basics. There are very complex domains and very basic domains, but the following list should fit any of them.

| | |
|---|---|
| Visualization: | Support your statements with drawings, figures etc. |
| Examples: | Use examples from another domain that everybody knows |
| Attributes: | Use physical attributes to support your explanation |
| Clock: | Make sure you can always explain everything within 2 minutes |

I hope that this will help to keep our customers in line with testing.

## > biography

**Thomas Hijl**
*is acceptance manager and partner at Qwince BV. He started in 1996 as consultant / project manager for Philips Semiconductors (a.k.a. NXP) working on technology projects in the EMEA, APAC and AMEC regions. He switched to test management because of his ambition to improve project deliverables. He now serves clients with managing acceptance throughout the project lifecycle.*


# Testing @ shipping domain

*by Reshma Zilpilwar*

Domains are usually not emphasized enough and instead the concentration is directed to applications, which is not quite the right approach. Testing with knowledge of the applications under test is not enough. What is needed is thorough domain knowledge, also in order to do justice to the quality and integrity of the application. Some common features can, of course, be tested well without domain knowledge, but getting all the kinks ironed out definitely requires a thorough knowledge of the related domain.

I believe the following approach would help in the testing of domain-specific applications:

1. Learn the domain
2. Understand the business
3. Know the application under test

Disadvantages of not knowing domain knowledge could be the following:

### *Incomplete test coverage*

When I first prepared logical scenarios before knowing about the domain, I could hardly gather more than about 25-30 scenarios from what I could see on the screen. However, after getting to know about the rules and the way business is carried out in shipping, the number of scenarios rose to more than 110.

### *Delay in testing*

If a person is to test a banking application (BSFI) and is unaware of the way day-to-day work is carried out in a bank, and of the terminologies used (such as cheque book, passbook, debit, credit, etc.), then he/ she may end up consuming more time understanding and then applying the knowledge to test the application compared to a person who already has some knowledge of it. Such a person can be more productive and quicker than a person with zero domain/industry knowledge. So domain knowledge adds value to a project/product.

### *Unable to step into end client's shoes*

A tester lacking the end-user perspective would not be in a position to contribute to improving the look & feel as well as identify more bugs on the UI front at initial stages of the project.

### *Lack of understanding the functional/technical flow*

Without domain knowledge it would be difficult to understand the functional flow & business rules which would help us understand the product requirements in a more businesslike approach. It wouldn't give him/her better understanding on back-end processing: how effectively/ efficiently the data / code is handled and how it could be used to generate more scenarios for testing.

### *Defect triage*

Domain knowledge is also important for defect triage: Knowing how the application will likely be used and how it is expected to perform will tell QA whether a given defect is trivial, significant, or critical.

### *Difficulty in getting well versed with terminologies*

Knowing the business terminologies is also important to get comfortable in understanding the requirements, especially in communication with the client. Reporting in the language of the business will result in good rapport with the business team.

The shipping domain doesn't sound complicated unless you actually get into the applications. In one of my projects I was asked to test an application used for shipping chemicals. At first glance, I found the application quite simple and easy to understand, but when I actually took a step towards actually writing the logical test scripts, all I could get was a blank. Things became much clear when we learned the domain / business knowledge.

Shipping of chemicals mostly involves some set rules which need to be followed, e.g., a particular chemicals is not allowed to be shipped in particular containers due to its chemical properties like acidic nature etc. Getting to know the exact rules was not so important, but to get an idea of these rules and how we could use them in our testing of the application was of more importance.

There was another case which I came across later and would like to mention: A particular chemical has some description associated with it, which needs to be mentioned in the final certificate issued to the carrying ship.

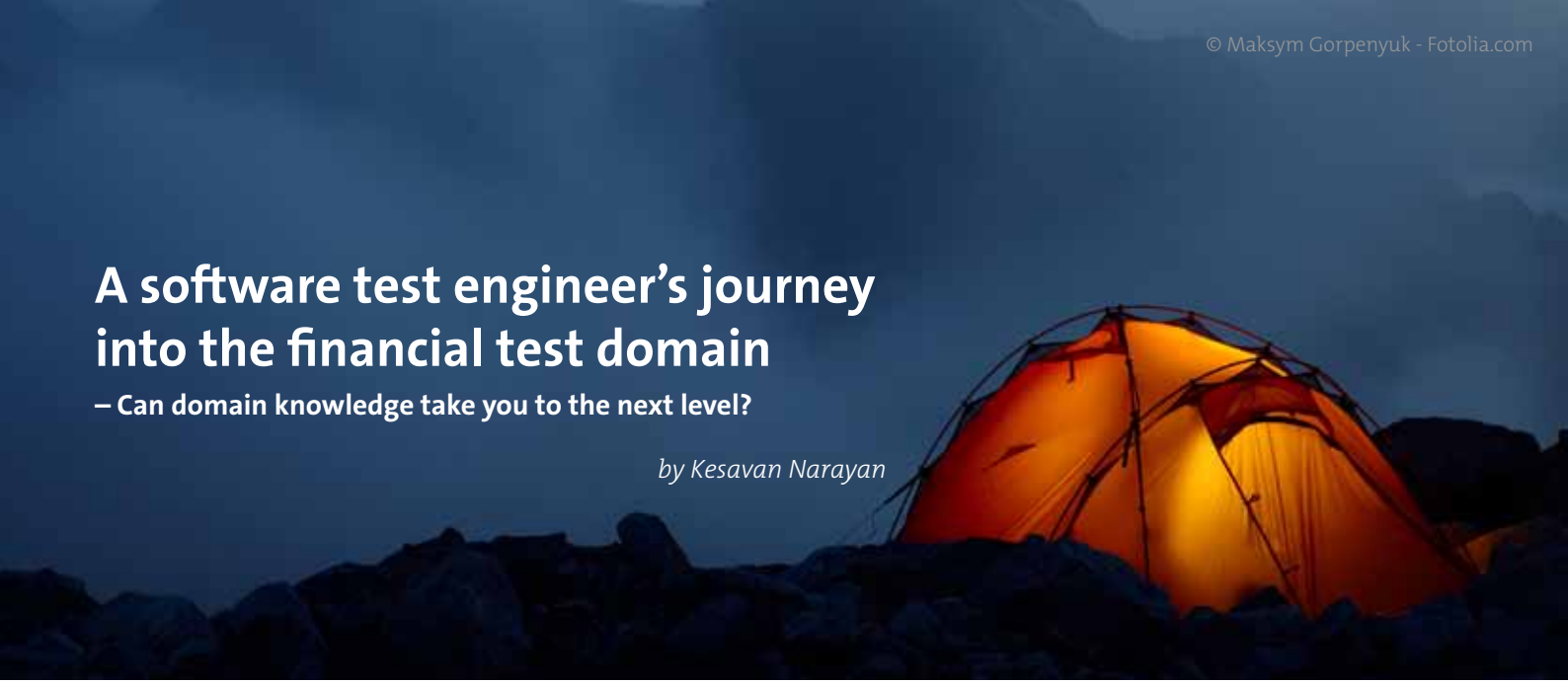
Before entering the shipping domain, the following points would help in understanding the domain:

- Vessels are nothing but ships.
- Tanks are nothing but containers on ships in which chemicals are shipped.
- Every ship in the UK is uniquely identified with an IMO number, which is provided by a central authority.
- There are many types of cargo ships like bulk carrier, tankers, multi-purpose cargos etc.
- Each chemical has certain properties. Depending on these, it could be shipped in a particular tank group which also has certain properties.
- At the end of all this survey, a certificate is obtained which acts as a permit for shipping chemicals.

Hence, whenever there's a need to test any specific domain application, the first step should be learning the domain and then understanding the business. Knowing the application could then follow. Domain knowledge also gives an understanding of functional flow which, in turn, helps us improve the quality of the application and increase customer confidence.

## > biography

**Reshma Zilpilwar**
*has around 7 years experience in manual testing with QTP & Oracle ATS awareness. She also has ISTQB & ITIL certifications which give her exposure to various testing & service line methodologies respectively. Her experience in domains varies from finance to retail and from shipping to service line.*




# A software test engineer's journey into the financial test domain

## – Can domain knowledge take you to the next level?

*by Kesavan Narayan*

This article is an attempt to encourage software test professionals to think about "Test Domains" and find a tangible connection between valuable test domain experiences and incorporate it into the overall software test strategy to help deliver products that will ultimately result in a high level of quality and delighted customers. I'd like to share a "story" of my personal journey to uncovering the importance of "Test Domains" with my experiences in the financial software domain.

The fact that domain expertise in the product's domain is important when building software would seem like stating the obvious to software test professionals. However, it's the sharing the journey and exploring the rationale for such reasoning that I find more interesting and perhaps thought provoking. If you as a software test professional are looking to improve your value to your software development team, you must become familiar with your "Test Domain" and learn how to incorporate that knowledge into your overall test strategy.

### *Test Domains: What does it really mean?*

Let's take a closer look at the term "Domain Expertise" to help set some context. Domain expertise typically implies that an individual has some practical experience in the given area/field and in some cases the educational background in the "domain" to complement domain knowledge acquired by experience. Therefore, experience in the field and educational background in the field can be considered the two components that contribute to "Test Domain Expertise." In the case of testing software, we have a few tenants we see in any software vertical when it comes to delivering innovative software or services to our customer:

a) Understand the problem you are trying to solve

b) Understand the customer's pain points and the domain

At a high level, that's what we are looking at, irrespective of any domain we choose to test software in with the constant objective of delivering a high quality product.

Given the context for what I'll consider "Test Domain Expertise" we then come to the following question: "Is domain knowledge required for a software test professional to be successful?" Could the software test professional's consideration of their test domain perhaps make their work more valuable and efficient? I strongly believe that "Test Domain Expertise" is essential in software testing if one desires to reach the highest level as a software test engineer in any product domain.

### *A not so travelled path to "Test Domain Expertise"*

At 16 years of age, I worked hand in hand with my father at an East Providence, Rhode Island Gas Station – I was a typical small business owner's son doing many things at an early age I would seldom realize the impact of how it would potentially lay an imprint on my future or career. Besides providing me with a solid appreciation for hard work, I never felt that this job of helping in the running of a small business would ever be reflected upon in my future as a test domain. I thought the small business experience at the gas station to be more of a noble attempt to play my part in supporting my family in the cost of an education as an engineer. I never imagined that this experience of assisting in running a small business would lay the foundation of an experience that would resonate so well in my career.

Upon spending my first 3 years at Intuit on an advanced platform team and buried in a technology stack far away from the customer, I learned and watched a culture that is rich in process and methodology when it comes to understanding your customer at a detailed level. From that team, I moved onto the QuickBooks Online® team – providing accounting software to assist small businesses to succeed. The QuickBooks Online® team provided an opportunity to step into an arena where test domain expertise was very critical to success.

Once on the QuickBooks Online® team, many lessons I learned by experience in the small business domain resonated very well when considering how to improve, test and think about our product. As a software quality engineer initially joining the QuickBooks Online team as a technical resource, the challenge became clear to me that expanding upon the technical skill set with experience from the test domain would help increase my value and effectiveness to the team. Many years spent working at a gas

station along with managing the business and living the small business culture would now become irreplaceable test domain knowledge.

The lessons learned about the challenges that must be overcome for small business owners to succeed helped pave the way for deep customer empathy that's required when considering the overall value of our company's service. Equally important was considering this component in our overall assessment of the quality of our product. I found myself thinking back to my days working at the gas station, maybe not so much to filling gas, checking the oil, and checking the tire pressure, but shifted my focus to the times when I did the daily accounting by hand, took an inventory count and tracked it on paper and calculated the daily balance sheet based upon sales and purchases. The experience of running a small business satisfies one of the two components I discussed earlier as a classification of a "Test Domain" – first-hand experience in the given field.

### *Taking the software test engineer to the next level*

How does somebody with knowledge in a given test domain transcend to the next level as a software test engineer?

There are some basic testing components we all as software test professionals should be aware of, but sometimes, like in sports, the "fundamentals" are worth reviewing. In our case, the fundamentals of software testing components that make up a comprehensive test plan can consist of the following:

- Real-world testing or system testing
- Usability testing
- Functional testing
- The "illities" (scalability, security, vulnerability and performance,....etc).

Knowledge in a specific test domain can enhance every single test you develop. This is more so in the areas of usability, functional testing and what some call "real-world testing."

"Customer empathy" is a critical factor that experience in a given domain brings to software test engineers. This state of mind is an intangible factor that can't always be learned and must be "experienced." In the case of the small business domain, I look for areas to improve our product from the view of a small business owner based upon my personal experiences in the domain along with actively monitoring the domain. The state of mind described requires domain experience or an ability to seek it out and often results in benefits such as improving your product as well as making your overall test strategy more effective.

Consider "Customer Modelling" – the methodology used to produce the type of test cases that were just described by considering inputs such as the product domain and the workflows your customers use in your product in the given domain. We'll explore this concept further in context of the QuickBooks Online® team's test practices considering "Domain Expertise".

### *The QuickBooks Online® team's approach considering domain expertise:*

Our test domain is financial accounting software for small businesses. To be more specific, we need to understand how our small business owners work, assist them with optimizing workflows to produce accurate financial accounting of day-to-day business transactions. Our software service must help small business owners save time and money and help run their business successfully. The team is always evolving techniques to keep a tight connection with our test domain due to the critical role it plays as an input to our overall test development and test strategy in producing high quality software.

### *Test domain information framework:*

In a world that is very dynamic when it comes to customer feedback, we've done several things to keep our "Test Domain Expertise" fresh for engineers on our team. The QuickBooks Online® test team worked to build a "Test Domain Information Framework." The framework is designed to help infuse the test team with knowledge of our specific "Test Domain." It is also essential in bringing some key mindsets and concepts to our team such as "customer empathy" and "customer modelling." The framework is pretty simple - all it requires is for your team to spend time to think about your specific test domain and the resources that would provide actionable information about your customer operating in your product's domain. Consider that the "Test Domain Information Framework" produces content that cannot only enhance your test scenarios but also bring forth tests that may never have been considered at all.

In practice, the QuickBooks Online® team has engineers "monitor" various feedback sources in the domain and from our customers for constantly changing information specific to the small business domain. These resources are rich with new trends in the small business domain considering new workflows in using our product, accounting challenges, and pain points small businesses face in managing their business and finances on a day-to-day basis:

- In product customer feedback mechanisms
- QuickBooks Online® blog
- QuickBooks community forum
- Engineer designed key Google alerts in small business accounting
- Twitter
- Customer contact programs
- Review of customer support issues

By monitoring such a set of dynamic resources that so intimately ties our customers to our test domain, this becomes a vehicle by which our team can acquire the "customer empathy" mindset. This mindset is what is required by the software test professional to test beyond the feature by considering the usability, domain specific practices, and overall value of the product to the customer in the domain.

### *Customer modelling*

"Customer Modelling" captures the essence of what the output is of monitoring our "Test Domain Information Framework." This concept captures the link we are looking for when attempting to connect test domain expertise to a concrete test strategy. For any software service or product intended for the end user, one needs to consider the fact that typical user interfaces allow for many paths to accomplish a set of "tasks" by the customer. Given the number of paths a customer may take to accomplish a task in an application, capturing the sub-set that your customer uses can be crucial in test case development efficiency. Once the customer-"influenced" sub-set of use cases are identified, the next step is to create a test set simulating those specific workflows and limit emphasis on other permutations that are less relevant or paths not utilized by the customer. The ability to succeed in this form of test case development hinges upon the level of test domain expertise you have or acquire. Developing test cases using this concept brings efficiency to your test development time line by producing test cases that capture your customer's intent and pathways in the product they utilize most. Consider the efficiency required when creating UI automation test suites - "time to execute test suites" is a major concern with this level of automation due to the overhead of navigating through the UI layer as opposed to the API layer. It is critical that the automated suite we create from our test cases is an efficient set of tests due to the limited time available when executing UI test sets in a continuous manner (CI). Test case efficiency is critical as software test professionals are always confronted with a shrinking test window due to faster "time to market" pressures.

### *Test domains: Utilizing the test domain knowledge in a concrete QuickBooks Online® example*

Let's use the example of testing the QuickBooks Online® inventory costing component. This use case is interesting in that we have a test domain that not only deals with general accounting for small businesses, but it also introduces the challenge of inventory cost accounting which is a more advanced concept only "product based" businesses need to consider. In this case, my experience at the gas station was a good reference point, but I had to delve deeper into a small business concept that was new to me. Given my test domain experience was more simplistic in nature and not a pure "product based business," I had to leverage one of the techniques I mentioned to acquire domain knowledge: self-educate yourself in the test domain. I looked to a text book on inventory cost accounting and internal domain experts to educate myself further in the test domain. Once you infuse your test cases with domain knowledge, it's at that point when the core software engineering skills such as developing automated tests, performance use cases and functional tests for basic acceptance criteria become so much more powerful and effective.

## Test component: Inventory cost accounting using FIFO (First In First Out)

### *What was the test strategy?*

We start by understanding the nuances of our "Test Domain" for a "product based business" using inventory cost accounting. Look to the "test book" use case in the inventory cost accounting literature. Collaborate closely with the lead developer that happened to also be an accounting expert to obtain the "customer empathy" mindset and understand the challenges in managing inventory for small businesses. Use this mindset as a basis to partner with the lead developer to not only consider inventory cost accounting principles to exercise the product, but also valid customer scenarios driven by understanding the customer's challenges in the domain. Finally, re-use test cases from an inventory cost accounting module that Intuit had already developed for our QuickBooks Desktop® product. The re-use of the existing inventory cost accounting use case also required further investigation of the domain to modify for the change in inventory costing methodology (average costing vs. FIFO). In this case, basic business transaction sets were re-used from the QuickBooks Desktop® test cases for our QuickBooks Online® inventory costing module, and the financial data required was recalculated for numerical correctness considering the FIFO method.

### *What were some best practices leveraged?*

Test plan reviews with the lead development engineer/accounting domain experts were conducted iteratively. We introduced knowledge gained by the previous test engineer that worked on inventory cost accounting products at Intuit into the Test Plan. Inventory cost accounting text book use cases were re-worked and replicated for use as baselines for expected detailed accounting

outcomes with our inventory costing engine. We spoke of the "customer empathy" mindset in addition to "customer modelling" as techniques to help build a richer test set. In this specific test case we see how both concepts can create an effective test: Based upon domain knowledge we anticipated counting errors by small business employees in obtaining inventory counts (it's a common occurrence), so we consider the concept known as "shrinkage" in inventory costing and develop test cases for such conditions that the customer may face. Accounting practices of businesses such as "closing the books" and how that would impact the addition of inventory costing module for existing customers when first adding it to their existing account were considered. We built a solid set of test cases by utilizing our domain expertise in inventory cost accounting and practicing good peer reviews for all test assets, including test plan, test cases and overall test strategy.

### *What was the outcome?*

The engineering team produced a successful set of use cases that traversed the various situations a "product based" small business requiring inventory cost accounting would encounter. The use cases/ test cases acted as the "acceptance" test set when considering the more complex challenges for the software component to operate against. Regression effort was streamlined as the test cases were an efficient set that captured our customer's intent considering the test domain. Though the use cases were manually executed at the time, these tests introduced a rich business context that could help improve our UI test automation suite down the line with efficient test cases considering the time cost of executing UI automated test cases.

### *How did we do post-release?*

After the production launch, the inventory costing accounting module in QuickBooks Online® was well received by our customers and released at a quality level that was excellent (o high impact post release issues reported by customers). The high level of quality exhibited by the low defect escape rate in production can be attributed to the careful consideration of "domain knowledge" into our overall test strategy. As far as the domain was concerned, this was a team effort that started from the lead developer, quality engineer to the entire cross-functional team. We've stressed the benefits of domain knowledge for the software test engineer but the most successful products maintain domain knowledge cross-functionally.

### *Considering "Test Domains" for your next project*

When considering your test strategy for your next project, consider your specific test domain to ensure the best outcomes for the software service or product you present to the customer. It does not matter if you happen to acquire test domain knowledge from experience or educate yourself in the domain. In the examples of inventory cost accounting, domain knowledge acquired by experience was invaluable to assist with "customer empathy", but further "text book" learning and collaboration with inventory costing domain experts was required to implement a successful test strategy and produce solid test cases. The key value of considering the test domain by using some of the concepts reviewed is the ability to have utmost confidence in the resulting test cases produced, which is something all test professionals seek.

## > biography

**Kesavan Narayan**
*has over 15 years experience in software quality assurance and has worked on various products spanning a variety of domains from Computer Aided Design (CAD/CAM) software, web performance management, and more recently small business financial accounting software. Kesavan has a passion for research into Agile software development methodologies that consider test developer productivity and unique ways to produce high quality software through short iterative development. Currently, Kesavan is working at Intuit, Inc. on the QuickBooks Online® Team in Mountain View, CA.*


# Automotive and medical: A comparison of IT standards for safety-relevant products

*by Dr. Anne Kramer*

In the automotive industry special attention is being paid to the upcoming standard ISO 26262 that is currently been finalized. ISO 26262 defines all activities of the so-called „safety lifecycle" for electrical and electronic systems in road vehicles and implements the domain-independent standard IEC 61508 for the automotive industry.

The introduction of this new automotive standard is driven by a major concern regarding safety of the user. A similar situation can be found in other safety-critical domains, where development processes are also highly regulated by laws and standards. We would therefore expect that the activities prescribed or recommended by the regulations are more or less the same in all safety-relevant domains, be it medical, pharmaceutical, automotive or avionics.

Despite this, only little exchange takes place between the medical and the automotive domain. This is mainly due to different vocabulary and assumptions. In this article we compare the automotive standard ISO 26262 to the medical standard IEC 62304, pointing out the most striking similarities and differences, and in doing so try to bridge the gap.

### *Scope and level of detail*

The most striking difference between the two standards is their number of pages. The nine parts of ISO 26262 together are nearly twice as thick as IEC 62304. This has two reasons:

First, the two standards do not have exactly the same scope. ISO 26262 is strictly focused on safety-related systems and not limited to software. The standards for medical devices apply to the entire product, but require more stringent tests and documentation for safety-relevant systems. Of course, standards for hardware development of medical devices also exist (but have not been taken into account in this paper).

Second, the level of detail is very different. While the standards for the development of medical device software are kept rather general, leaving it up to the manufacturer to find and justify appropriate processes and measures, ISO 26262 gives detailed instructions on the way the process requirements shall be implemented. Curiously, this also influences the way the standards are read and interpreted. The major difficulty in the medical domain lies in the question: „What exactly is expected from me?" With the standards being extremely succinct, each word is taken as a requirement. ISO 26262, instead, contains a large number of recommendations, e.g. concerning test methods. Thus, it is rather interpreted as a framework.

### *Vocabulary and philosophy*

Another important difference between the two domains is the vocabulary. This makes communication and the interchange of information, approaches, methods etc. difficult. The highest confusion is created by identical terms with different meanings. For example, „SOP" in automotive stands for „Start Of Production" (instead of „Standard Operating Procedure" in the medical domain). Also, the term „system" is used differently. ISO 26262 defines a system rather technically as a „set of elements, at least sensor, controller, and actuator, in relation with each other in accordance with a design". IEC 62304 (as well as ISO/IEC 12207) defines a system as a composite of procedures, hardware, software, facilities and personnel providing functionality to perform a given task or achieve a specific purpose. The second definition is broader, giving the term „system validation" a completely different meaning.

The highest similarity between the two standards is the common underlying philosophy which is a risk-based approach. The degree of documentation and testing required depends on the risk that someone might get severely hurt by using the end product. Both ISO 26262 and IEC 62304 require a hazard analysis and – as a result of this analysis – the classification of the system regarding to its safety relevance. However, the classification differs in some details. IEC 62304 distinguishes three classes ranging from class A (no harm or injury possible) to class C (severe injuries or death possible). ISO 26262 classifies the systems according to their „Automotive Safety Integrity Levels" (ASIL), where ASIL A corresponds to the least stringent and ASIL D to the most stringent safety integrity level. The classification criteria in the two domains are shown in the tables.

Another similarity is the notion of inheritance of classification. If medical software is split into modules, these modules inherit the classification of the ensemble unless a rationale can be provided

that a module may be classified differently. For automotive systems, sub-elements inherit the ASIL of the entire element. ASIL decomposition is allowed (and even encouraged), but it is subject to specific rules that are defined in part 9 of ISO 26262.

In part 2 of ISO 26262 it is stated that „The system development process is based on the concept of a V-model (...)". This does not necessarily exclude agile approaches, but adds a formal framework concerning the traceability of requirements and the chronological order of design, implementation and testing activities. In the medical domain, the approach to agility is the other way round. The V-model has been the most widespread lifecycle model in the past. Nowadays, agile approaches are getting more and more popular. Still, the fundamental requirements of traceability and chronological order have to be respected within each sprint.

### *Verification and validation*

Essentially, the requirements for verification and validation activities are identical. Tests shall be specified at all levels of integration. All activities shall be planned and documented. Tester qualification must be proven. Documented reviews are systematically required and last, but not least, requirements traceability is extremely important.

Again, ISO 26262 is much more detailed than IEC 62304 on the methods that should be applied to derive test cases and which type of tests (e.g. requirement-based tests or fault injection tests) should be performed. Several tables list recommendations depending on the system's ASIL, and detailed instructions are given concerning tool qualification.

The major difference is related to the different definition of the „system". In the medical domain, the product has to be validated together with all other system components. These may be, for example, some off-the-shelf software in the hospital. In ISO 26262 system validation is required at vehicle level of integration. This includes user test under real-life conditions performed by normal users as testers that are not bound to prior specified test scenarios. Similar to the medical domain validation at vehicle level shall be based on safety goals, functional safety requirements and the intended use (operational use cases). However, usability tests that are an important part of medical device validation are not explicitly addressed in ISO 26262.

For medical devices, usability is considered as being safety-critical, since it might be life-threatening for a patient if the doctor cannot see a warning or cannot reach an important button in time. Also, the understandability of the user manuals is taken very seriously, because it may become safety-critical if a procedure is documented in a wrong or misunderstanding way. In the automotive domain the user interface of the vehicle is limited to few elements in the car's dashboard. All devices with more complex user interfaces (such as navigation systems) are considered as extras that are not part of the vehicle itself. In case of an accident, the responsibility is clearly with the driver. This is not necessarily the case in the medical domain, where the manufacturer may be held responsible for accidents due to a lack of usability.

In the automotive domain, therefore, usability may be important as a selling argument, but it is not a safety issue. However, with the continuously increasing amount of safety-related functionality in the area of driver assistance and vehicle control, the driver relies more and more on this support. It is an interesting question whether usability issues will become more important for automotive safety in the future or not.

### *Conclusion*

We compared standards of two safety-critical domains: medical and automotive. As can be expected, a large number of activities coincide, since they simply correspond to common best practices that can also be found e.g. in ISO/IEC 15504.

Nevertheless, cross-domain communication remains complicated due to differences in the vocabulary and interpretation of the standards. Being more detailed, the automotive standard ISO 26262 could provide guidance, e.g. for tool qualification. On the other hand, usability aspects that are already taken into account in the medical domain might become of interest for vehicles (in particular for driver assistance) in the future.

We believe that great synergy may be obtained by a more standardized approach to software development. The exchange of experiences and methods across all safety-critical domains should, therefore, be encouraged.

| Classification of medical device software | |
|---|---|
| Previous hazard analysis according to ISO 14971 taking into account severity, probability of occurrence and (optional) probability of detection. | |
| Class A | No harm or injury to the health of the user, the operator or any other person can be caused (even if the software does not work as specified) |
| Class B | No major harm can be caused by the medical device.<br>Major harm is defined as injuries or diseases that might directly or indirectly lead to death, result in an irreversible handicap or require medical or chirurgical interventions to repair the damage. |
| Class C | Major harm or death is possible. |
| If hardware mechanisms are implemented to reduce the risk caused by software, the software classification can be reduced by one class. | |

**ASIL determination of automotive systems**

Previous hazard analysis according to ISO 26262 taking into account

- Severity (S1 = "light and moderate injuries" to S3 = "survival uncertain and death"),
- Probability of exposure (E1 = "very low" to E4 = "high") and
- Controllability (C1 = "simply controllable" to C3 = "difficult to control or uncontrollable).

ASIL A to D, depending on the results of the hazard analysis:

| **S1:** light and moderate injuries (S1) | | C1 | C2 | C3 |
|---|---|---|---|---|
| | E1 | - | - | - |
| | E2 | - | - | - |
| | E3 | - | - | A |
| | E4 | - | A | B |

| **S2:** severe and life-threatening injuries (survival probable) | | C1 | C2 | C3 |
|---|---|---|---|---|
| | E1 | - | - | - |
| | E2 | - | - | A |
| | E3 | - | A | B |
| | E4 | A | B | C |

| **S3** survival uncertain, fatal injuries | | C1 | C2 | C3 |
|---|---|---|---|---|
| | E1 | - | - | A |
| | E2 | - | A | B |
| | E3 | A | B | C |
| | E4 | B | C | D |

Hazards that are estimated as S0 ("no injury"), E0 ("incredible") and C0 (""controllable in general") do not require ASIL determination.

## > biography

**Anne Kramer**
*works as project manager and senior consultant at sepp.med gmbh, a service provider specialized in IT solutions with integrated quality assurance in complex, safety-relevant domains.*
*Anne holds a diploma in Physics and received her PhD at the Université Joseph Fourier in Grenoble (France). Her first contact with quality assurance dates back to 1996, when she started her career as a software developer for smart card testing tools at Schlumberger Systems in Paris. Two years later she became project manager for point of sales terminals.*
*In 2001 she joined sepp.med where she specialized in quality assurance methods and IT development processes for the medical and pharmaceutical domain. She regularly contributes to magazines and conferences as author or speaker. Anne holds a certificate as Certified Professional for Requirements Engineering (Foundation Level).*

German Testing Board

www.german-testing-board.info

# ISTQB® Certified Tester

## Über 155.000 Certified Tester weltweit. Wann gehören Sie dazu?

## Manche Ausbildungen öffnen Wege, andere eine ganze Welt.

Die Schulung zum ISTQB® Certified Tester führt zu einem weltweit anerkannten und etablierten Zertifikat. Rund 17.000 Fachkräfte alleine in Deutschland haben es schon. Und Zehntausende in weiteren 47 Ländern - von A wie Amerika bis V wie Vietnam.

- Der ISTQB® Certified Tester ermöglicht Karrieren - mit ihm liegt erstmals ein internationales und branchenübergreifendes Berufsprofil für Software-Tester vor.
- Der ISTQB® Certified Tester macht die Arbeit leichter - Tester sprechen nun die gleiche Fachsprache, benutzen die gleichen Begriffe.
- Der ISTQB® Certified Tester hilft Kosten zu senken - Durch geschulte Tester werden Fehler bereits in frühen Phasen der SW-Entwicklung entdeckt.

## Anmelden ist einfach.

Ein akkreditierter Trainingsanbieter ist sicher auch in Ihrer Nähe:

### GTB Premium Partner

- ALTRAN GmbH & Co. KG
- andagon GmbH
- berner & mattner Systemtechnik GmbH
- Certitude GmbH
- corporate quality consulting GmbH
- EXCO GmbH
- imbus AG
- IT-Testing.de
- Knowledge Department GmbH & Co. KG
- Logica Deutschland GmbH & Co. KG
- MaibornWolff et al GmbH
- Method Park Software AG
- oose Innovative Informatik GmbH
- Philotech GmbH
- qme Software GmbH
- sepp.med gmbh
- Sogeti Deutschland GmbH
- SQS AG
- T-Systems

### autorisierte Zertifizierungsstellen

DLGI - Dienstleistungsgesellschaft für Informatik mbH

iSQI - International Software Quality Institute GmbH

Alle Trainingsprovider siehe www.german-testing-board.info



# Testing in the capital markets domain

*by Sarat Kumar*

Given the competitiveness that drives the Wall Street and other financial hubs, one can imagine how focused each firm would be to bring in product innovations leveraging technology. Such innovations would help them differentiate among the best of the best in the eyes of their clients. Add to that the troubles the industry has been going through towards the end of the last decade. Collectively all these have been creating situations such that, good or bad days, these firms need to be prepared for high volumes of trading. And they need to be fast and adept in processing what they get, and in delivering what their clients need. As a firm you wouldn't want to lose the trust of a client and lose the business to a competitor; trust is what you will find the hardest to win back. By and large, these are applicable to all industries and domains. So what is different in capital markets? – You find perhaps the competition the fiercest and mistakes the most pricey.

Now when the business units in such firms turn to technology to create or enhance their competitive edge in the market, they expect a high degree of total quality from the products developed. You would also find a true reflection of the competitiveness from the technology divisions , where each technology division aims to develop the best of the algorithmic engines, DMA platforms, smart routers, dark pools, and so on.

The demanding nature of delivery excellence is easily passed on to the testing teams. One needs not only to be very knowledgeable about the industry (and deep into the specific area of work), but also one needs to have that zest for acquiring more business knowledge. This enables you to do your job well and puts you in a position to offer more added value. We'll look at some of the nuances of testing and related aspects in the trading technology domain, mostly from the viewpoint of a sell-side trading firm.

### *Regulatory changes and other high priority projects*

Regardless of whether they are driven by external factors (like regulatory changes) or internal factors (from business or technology), projects in this domain are usually run under very tight deadlines. As is the case of any large size business in the modern world, the capital markets domain is also heavily dependent on technology. So one would imagine that the feasibility of technology implementation would be one of the key factors in deciding the timelines; but at the same time you would experience that it is definitely not *the* most important factor.

How do you deliver for something extremely critical in nature, which has a large number of business scenarios to be covered, and within much less time than is actually required? You are being forced to be creative here, and that's what you will need to be. You need to make sure that sufficient test coverage is provided while the risks identified are minimized.

There are accelerators (like test automation) that will help you achieve some of these goals. But there could be situations where it will be tough to decide whether you even have enough time and resources at hand to implement the accelerators. A great in-depth understanding of the requirements, as well as the ability to use that understanding for testing and risk analysis, would provide you with the biggest return on investment. Let's see what specifically can help you achieve this understanding.

### *Domain understanding improves the effectiveness and efficiency of testing*

In trading, at a very basic level, what you deal with are orders that you receive from clients and then executions from an exchange (or a similar destination - internal or external) at the best price as quickly as possible. There are many attributes associated with an order and corresponding execution messages. There could also be multiple actions possible on these (though practically not many happen on executions). So achieving a full coverage of attributes and their possible multiple values (e.g., orders can be of the type market/limit/stop, etc.) will not only be difficult, but also an unwise utilization of man-power and other resources. If one understands the trading domain very well, it will be easier to analyze the change or requirement in hand and then determine *how much to test*. This domain, with so many message attributes and their values, is one of the major adopters of the combinatorial value-pair approach for testing. It is very much applicable to automated tests as well as manual functional tests, and provides the right balance between time-in-hand and test coverage.

With the greatly dynamic nature of the capital markets, the testing methodology that works very well from an efficiency standpoint is Agile. So it is essential that the tester speaks the trading markets language with the various teams he/she will need to interact closely with, such as application developers, business users, and so on. One thing you will find tough is the fact that the level of knowledge required to work in such a demanding environment is not easily built in a short span of time. As a tester, one needs to be really proficient on the domain aspects.

As a tester you will, of course, also be expected to find defects of a technical nature; but it is imperative that you think of the business scenarios that could be of potential risk. For instance, if you are a front-end tester for a derivative trading application, you will need to think of many more scenarios which are specifically applicable to derivatives compared to those for a common

shares application. You also need to know what specifically you will need to verify apart from what you see in the GUI. There are several details relevant for downstream processing (such as trades booking, allocation) which may not be available in the UI. So the understanding of such dependencies will be very useful when certifying the application changes as production-ready. Similarly, having a good understanding of the domain helps you immensely in implementing a risk based testing approach, when necessary. As mentioned earlier, testing all possible combinations of attributes and values may be humanly impossible (and even using scripting for automation), but strong domain knowledge will help to assess the requirement and suggest what order types or execution instructions or settlement details are relevant to be verified for a particular change. This helps to reduce test execution time as well as the time for results verification and analysis.

### *Dealing with clients and market*

Another important fact to understand from a sell-side firm perspective is that each client may have their own ways or patterns of trading. Remember that the firms develop their own proprietary algorithms and products to outperform competitors. Each one of them will want your firm to provide them customized service. Then there are special days which are dear to the industry (like Triple Witching day or Russell Index rebalancing day). If trade processing is not accurate, these days can cost your firm dearly. So it makes the life of a tester more challenging in this domain because they need to know the patterns and other nuances of clients' trading, and to also know about those special days to be included in the tests.

What I'm alluding to is the key aspect of "knowing what happens in production". Until not so long ago, the life of a tester was pretty much confined to 'test against the given requirements and certify'. However, this has by now evolved to something of a more holistic nature. You as a tester now "own" the quality of the software while certifying it as "fit for production use", both from the functional and non-functional aspects. In this evolution it has become necessary for the tester to build a good rapport with the production plant maintenance team as well. Production teams will be more knowledgeable about special scenarios that occur in production and also how special processing days are handled. So they will be of invaluable help in keeping your firm's reputation intact and also yours as a great asset to the firm. Another significant factor that you may not have visibility in the test environment is about the configurations found in the production environments and the size/layout of those environments. Knowing this will help you design your tests in such a way that your test scenarios are as realistic as possible. It will help you in your risk analysis and allow priorities to be placed on the cases that are most likely to happen in production.

### *Non-functional testing aspects in the capital markets world*

It is very important to understand the specifics of non-functional aspects with respect to the domain of your work. For example, 'performance' may be the buzzword in non-functional testing discussions. However, it is important that the tester is sensitive to the fact that 'performance' has different perspectives, even within the same domain, when it is dealt by different stakeholder groups like application developers, QA teams, production plant maintenance teams, project managers, business users and so on. So it is imperative that the QA management makes it very clear to all interfacing groups what is going to be benchmarked in the non-functional area.

It is also important to translate the benchmarked numbers in such a way that they can easily be understood and interpreted uniformly by the above-mentioned stakeholder groups. They may need to make critical decisions during an event or a crisis based on these numbers. For simplicity, let us assume that events are externally driven and a crisis is an internal incident. For example, assume an exchange (say A) has some technical issue and wants the firms not to send orders to it. At that time, if your firm decides to route the orders to a different destination (B), the decision makers should be able to look at the current load going through the line handlers of "B", then look at the benchmark numbers / report and be in a position to take decisions very quickly. Typically, the time available to make such decisions will be much less during trading hours, and of course you don't want a bad decision or interpretation to affect the trading to destination "B" either.

Let's look at some of the key non-functional aspects from the plant operations perspectives, that are critical in the trading domain. One you would hear about just as often as performance is capacity. Typically, that means how many transactions, or, in other words, how many order-execution combinations, your application can handle before it slows down to below the acceptable limits or crashes (which you will never want). Here, while benchmarking the limits, the added value a tester can provide is the analysis of how much load balancing an application would need in production, based on the information of production trading volumes. Do tests with a scaled-down modeling of the production environment in your test environment. This will give you an advantage in testing the next major item: failover-recoverability. Of course, you wouldn't want "failure" of an application you tested in production; but you also know that you cannot be certain of this and need to make sure that the recoverability testing is also done and certified. Also in the trading domain, non-availability of applications or disruption in end-to-end flow during the trading window is an unimaginable situation from a criticality standpoint. Imagine the plight of a sell-side firm telling its clients that their business would be affected because of a system failure. Even a few minutes of downtime translates to very big losses in terms of dollars and reputation, so it is very crucial to know in advance that if something like that happens, your systems can be quickly brought back to normal service. Even when all systems are intact and processing, the edge you would get with your client over your competitors is the speed at which you can process their orders and get them back the executions at the best available price. So you will need to include isolated as well as end-to-end latency benchmarking in your testing. You need to provide this confidence to your business and client facing groups so that they can boast that your firm, with utmost certainty, can provide the best.

What will you primarily need to deliver all these to your stakeholders? Let's look at that next.

### *A quality testing infrastructure – not just a 'nice to have'*

A "world class test infrastructure", as my boss used to say, is one of the most important things you would require to produce a quality validation function, especially for the non-functional aspects. Until a few years ago in the capital markets domain, thoughts on QA environments was something like this: *You ask any business sponsor or application development manager, it would be one of their "good to have" items; you ask any QA manager, it would be his/her longing desire and a roadblock in demonstrating what QA can deliver.*

Even though a great QA environment enhances the quality of application delivery and, in turn, the business stability, it is strange that for several years this issue has not been looked upon as an important one. However, things have changed for good in recent years and I believe multiple factors led to this positive change:

a. QA organizations were successful in demonstrating small wins wherever good infrastructure was available.

b. Advancement of business thanks to technology (and thus dependency of business on technology) has increased exponentially. So not having well-certified software posed a great risk to the business.

c. QA organizations grew in maturity and were able to communicate the risks of not having an adequate infrastructure, and the added business value that comes with good infrastructure

As I mentioned earlier in this article, most of the high profile projects (whether regulatory or internally driven) are run with a very tight deadline in this domain. Now with that being unchangeable, how do you manage to get a well-rounded QA exercise done? For important projects of significant size, the QA execution will need to cover functional, non-functional (like failover-recoverability, capacity, latency tests, and so on), system integration tests, UATs, etc. With a limited timeline, you cannot afford to wait for a single environment to execute these steps one after another. So the solution is to build multiple environments where you can execute as many parallel streams of testing as possible.

One of the ways to efficiently utilize the hardware resources (and also push through the business case with the sponsors) is to explore virtualization, which will provide you with a significant cost saving. Your multitude of functional test streams can very well be executed in virtualized environments. However, you will need to verify with your application developers and production plant teams on using these virtualized environments for non-functional tests. Another important aspect to be covered from your side is to make sure that the environment you build with such high hopes also has extremely high availability. Aim for a near-production availability so that testers can use the environment as and when they need it. In the trading domain, you may want to include morning checks (possibly automated) that send some test orders and create executions so that you can confirm that your environment is in good health for the day. In this domain you will also need to test a lot of exchange/ECNs/dark pool functionalities. Typically, external destinations will have very limited test environments and even then the availability will be very limited. A good way to tackle this is to use simulators. There are good simulators available off the shelf and, if you prefer, you can build them as well, as the specifications of almost all destinations are readily available. This will greatly enhance the availability of your end-to-end environment and also give you the advantage of executing otherwise hard-to-test business scenarios.

How do we make a case for investment in good QA infrastructure? At more generic levels, how do we communicate the effectiveness of QA to the business/projects sponsors? Let's make that the next and final theme for now.

### *Reporting the right metrics*

It has always been a challenge to measure and project the effectiveness of QA to the business sponsors and other stakeholders. With the Agile methodologies and dynamic project environments, QA used to work hand-in-hand with various groups, but in isolated pockets. So in large organizations, it became very difficult for the QA management to communicate the positive impact of QA both to senior management on the technology side and to the project sponsors on the business side.

As the QA organizations evolved and matured, one method I've seen working successfully is to align the service of QA to the business sponsor groups (as verticals) and also to application groups (as horizontals). The reason to align things this way is that QA will be closely servicing various applications groups, which may in turn be servicing various business sponsoring groups. Along with this kind of model, make it a point to sit with the business sponsors and application owners during the planning phase of each project and application release, discuss what you plan to measure and report and get them to agree. They will have a lot of input to provide, which will help you meet your end goal. One discussion may not finalize the structure and details of QA metrics to report; but this process would lay down a strong foundation and you can incorporate any necessary changes as projects progress. (Of course, there could be other models that would work well in similar environments and that could vary from firm to firm).

The most important stakeholders may have only a small amount of time to scan through your reports and grab the details you want them to have. So structure the reports such that those items are at the top and presented as catchy as possible. Dedicate the top portion to an overall status indicating whether the project will meet the deadline or whether the deadline is at risk (and possibly 1 or 2 more such status categories). If there is a risk of not meeting the deadline, make sure that you have noted the main reason causing the risk also nearby. For example, one such reason could be "functional testing is behind schedule due to non-availability of test exchanges (caused by test infrastructure issue at XYZ exchange). In-house simulators are being explored". If the project is at risk, these kinds of pointers assist the senior stakeholders to pull the right strings immediately to help you remove the roadblock.

It is equally important to do a post-mortem analysis of the project/release and present this to the sponsors. Highlight the positive factors that helped you during the project as well as the areas for improvements so that your next project/release could be executed better for all concerned.

## > biography

**Sarat Kumar**
*has more than 13 years of experience in the IT field, mostly working in the banking and capital markets field. He is presently a Senior Test Manager with Infosys Technologies in India. He started his career in application development/ maintenance and shifted to testing and QA management half way through. He worked as a QA manager for a Wall Street major in the trading domain. His current areas of interest are test data management and tools for test automation.*

# Sie suchen das Besondere?

Díaz Hilterscheid

# Business Process Testing in Telecommunication domain

*by Hanoch Peleg*

The world we are living in has changed dramatically in the last 100 years, or should I say in the last 10 minutes?

In the good old days, fax and a landline were considered to be the state-of-the-art in communications technology, but now the last word has become the always-connected, always-online life style.

The demand for real time updates, the ability to make a fast connection, and the huge growth in desire for real time data and information, has totally changed the rules of the communication world in which we live. The luxury of enjoying this world, with the ability to access and use the new technology and exposure to "tons" of data in real time is always based on services provided by telecommunications.

The acceleration of software objects in telecommunications organizations, including the tight regulation aspects, has forced them to be more innovative and careful about the software testing performed in their organizations.

The life cycle of code development is becoming shorter as we speak. Whereas until now software version upgrades have been provided four times a year to support the new technologies, these upgrades are now expected to happen up to eight or ten times a year.

If in the past the software was developed as one big chunk, today's telecommunications system is built from tens of different systems and hundreds of interfaces. All of which must be able to talk to each other... in the same language!

The idea of testing systems individually no longer works, the systems have to be tested in an end-to-end flow to ensure that they work smoothly and in full harmony.

Having various systems connected to each other and interacting with other external systems, has created a new challenge for the testing organization. The permutations of flows in the systems are endless. The numbers of junctions and turns you can take in and out of the Telco systems has grown dramatically. Testing all the possible permutations has become unrealistic.

To avoid getting lost in Telco cyberspace, we at Amdocs SI have developed the end-to-end Business Process Testing Methodology, together with the tools that support and accelerate the reuse of our test scripts.

Before giving more detailed information about business processes, a few words about Amdocs SI Testing. Amdocs SI is an independent unit within Amdocs, currently supporting more than 25 top-tier customers, over 100 successful projects, with more than 1500 experts in over 20 locations worldwide.

For our customers in the Telco field, we provide a comprehensive suite of testing and complementary services. We have a proven track record of delivering complex projects worldwide 24/7 and "following the sun" projects.

Back to business processes - What is Business Process Testing?

The major change in the tester's mindset is that testing is no longer focused on the systems. Instead testing focuses on the way the organizations use the systems.

The tester now performs a process in the same sequence as it would be performed by the customer in production. Moreover, testing priority for business processes is determined by the extent of their use, as well as their effect on the enterprise, and on the telecommunication company's business targets.

Adopting this approach encourages us to focus on testing the important business processes, ensuring that our tests are covering exactly the areas that the client would want to be tested.

The question of efficiency is still pending, and we will also have to deal with that issue. Another change in mindset will be required: Efficient testing links a small number of essential business processes into a single end-to-end test scenario.

Hmm..., but where will we have the business processes laid out for us to pick, select and reuse? To this end, in Amdocs SI we have developed the Activity Library and the Calendar application.

Our experience in testing for large telecommunication compa-

# Wir auch!

Lassen Sie sich anstecken von der kollegialen Arbeitsatmosphäre in einem starken und motivierten Team. Zum nächstmöglichen Termin stellen wir ein:

**Senior Consultants**
**IT Management & Quality Services (m/w) für SAP-Anwendungen**
**Deutschland und Europa**

Sie haben

- eine fundierte Ausbildung oder ein Studium (z. B. Informatik, BWL, Mathematik) sowie mehrjährige Berufserfahrung als IT-Consultant in einem Fachbereich bzw. in der Organisation eines SAP-Anwenders, eines IT-Dienstleisters oder in einem Beratungsunternehmen in entsprechenden Projekten
- ausgewiesene SAP-Kenntnisse (z. B. SAP ERP, SAP BI oder SAP Solution Manager)
- Erfahrung im Customizing und mit ABAP-Programmierung
- Kenntnisse in der praktischen Anwendung von Methoden und Standards, wie CMMI®, SPICE, ITIL®, TPI®, TMMI®, IEEE, ISO 9126
- Erfahrung in der Führung von großen Teams (als Projektleiter, Teilprojektleiter, Testmanager)
- Vertriebserfahrung und Sie erkennen innovative Vertriebsansätze

Sie verfügen über

- Eigeninitiative und repräsentatives Auftreten
- eine hohe Reisebereitschaft
- gute Englischkenntnisse in Wort und Schrift

Dann sprechen Sie uns an – wir freuen uns auf Sie!

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).

Díaz & Hilterscheid Unternehmensberatung GmbH
Kurfürstendamm 179, D-10707 Berlin
www.diazhilterscheid.com

Mehr Stellenangebote finden Sie auf unserer Webseite:

- Senior Consultants Financial Services (m/w)
- Senior Consultants IT Management & Quality Services (m/w)
- Senior Consultants IT Management & Quality Services (m/w) für Agile Softwareentwicklung und Softwaretest
- Senior Consultants IT Management & Quality Services (m/w) für unsere Kunden aus der Finanzwirtschaft

Díaz Hilterscheid

nies shows that testing scripts for new functionality showed that up to 85 percent of the activities that were scripted to validate the new functionality were actually common activities.

Another survey showed that the same scripts are written over and over again for various testing processes that are performed for various items of new functionality, or for change requests.

In the telecommunications industry there are common activities that take a long time to execute and that are common to a number of business processes. A typical example is the "Run Billing" process. This is a batch process whose execution takes a long time and which runs for all the marked subscribers. For each user flow a separate customer flow is created, but the billing run activity will be performed once in a vertical aspect, for all the related customers.

At Amdocs SI the first step taken when approaching the planning stage of the testing process for a new release is to first follow the business processes, followed by a calendar type of test scenario.

The Calendar methodology was developed at Amdocs especially to support systems for testing telecommunications systems. This methodology is based on the fact that most of the business processes in this domain are based on events that are spread out over the day, week, month, and year.

Calendar is an Amdocs technique for testing processes in multiple parallel scenarios.

It is a two-dimensional matrix, in which the tested process appears as a set of activities in the rows and the multiple scenarios that are presented by the columns. The testing scenarios are various flavors of the same process that differ by using different tested data. The processes can be real, i.e., an actual business process, or they can be synthetic processes that were designed based on other aspects.

The scenarios are distinguished from each other by using different sets of data, processed using the same sequence of activities.

Here is an example that will clarify the situation: A customer purchases a mobile on one date, makes calls on various other dates, and receives a bill on yet another date. The customer price plans can change during that time, or expire, be upgraded, or enhanced, all on different dates. So this requires special attention to dates.

Consistent with this situation we have developed a stepped approach, and a set of supporting tools that allow us to create "Testing Calendars" - fast and efficiently.

**Step 1 - Mapping:** Map the business processes and then validate that the mapped business processes support the customer's business policies and strategies.

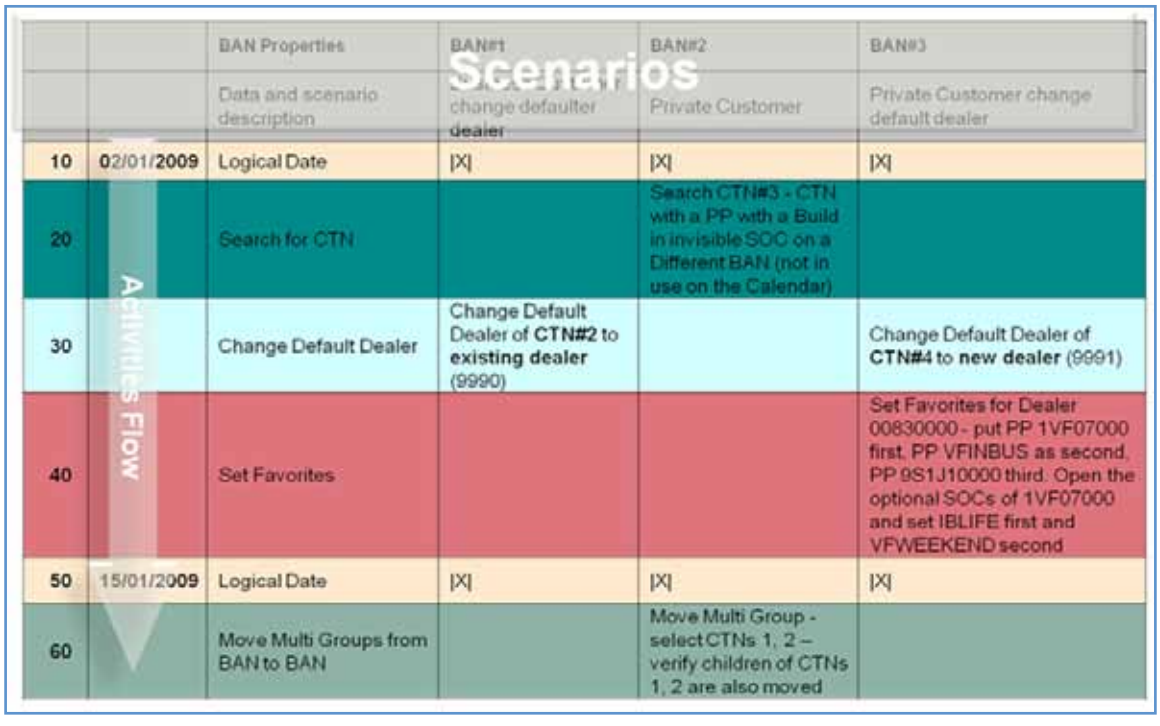
**Step 2 – Prioritization:** Prioritize the business processes that will be included in the testing scope, using a risk-based methodology:

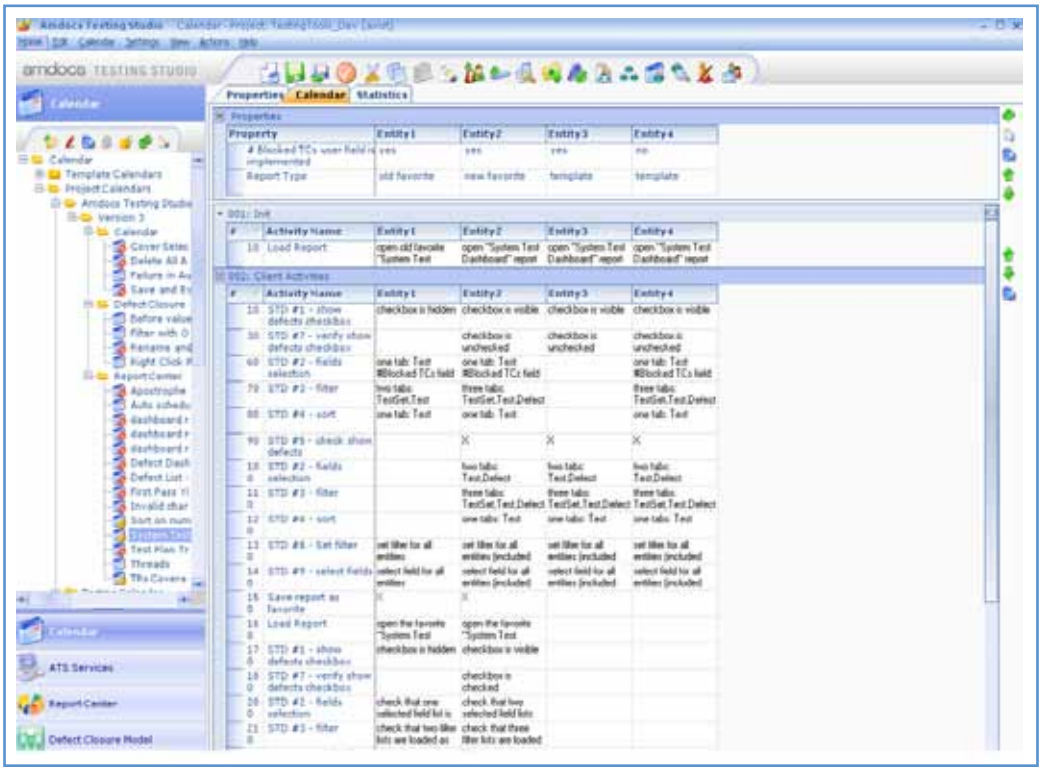
- Frequency of the business process
- The likelihood of this path being exercised in production
- How critical is the failure of this path in terms of time, money, security, customer, business impact

**Step 3 – Encapsulate in an "Activity Library":** Map the business processes into test flows:

- Identify test flows for each business process
- Each decision point in the business process diagram triggers another test flow
- Encapsulate the various business processes into a business process activity - A library that will be the repository for all the activities

**Step 4 – Detailed design phase:** Each business process test flow is translated into a list of sequential system activities, test cases (TCs), which are maintained in the Activity Library.





Once the above steps have been completed, we have reached the stage where all our business processes have been mapped and the supporting activities are in place, and everything is located in the Activity Library repository.

From this point forward we can mix and match any business process that we wish, without having to rewrite a script. Just select the required script and build the business process test flow by chaining the activities one to another. From now on we are reusing 100% of the effort that was invested when developing the Activity Library.

The next step is building the Calendar.

The calendar will be a logical chained business process that will imitate the days, weeks, months or years in the life story of a telecommunication company's customer.

In the old days, we used to build our calendars using the copy and paste method. Copy from the Activity Library area, into the about-to-be-born calendar...Well...not any more.

Today, we use a state-of-the-art Calendar application. Using this application allows us to build calendars with a new generation of GUI, in minutes.

Using a simple click and drop mechanism, the Calendar application allows us to build our calendars fast and accurate, and to ensure that all our business processes are covered. The Calendar application also maximizes the reuse and flagging for inefficient activities or syntax errors.

Once you have finished building your logical test scenarios, build on a logical component of business processes. Using just mouse clicks, an end-to-end script is prepared and ready for execution, including the detailed design script, representing the business flow.

Everything is tidy and ready for your tester to hit the keyboard and start testing.

By following the Business Process methodology, the strict Activity Library rules, and the state-of-the-art Calendar application, we have managed to reduce the testing design effort by 25 percent, increase the reuse to nearly 100%, and prepare the ground for efficient automation test development.

## > biography

**Hanoch Peleg**
*is QA Manager at Amdocs Israel Inc. He holds a Bachelor in Electronics and Computer Science from the Tel Aviv University.*
*Hanoch joined Amdocs Israel Inc in 2001 when he took his first testing management role for the Telco industry.*
*During his career he was responsible for the company's product testing and release processes, thus obtaining a wealth of knowledge in testing tools, automation, techniques and methodologies.*
*Hanoch has managed testing and integration projects in the Telco industry worldwide in the areas of billing, CRM, OSS, OMS, networking, mediation and device testing.*
*Hanoch is currently in charge of the method & procedure lead for Amdocs SI Testing for projects in the EMEA and APAC regions.*

# Handicap of a head start

*by Bert Wijgers*

**In order to test effectively we have to know how the application under test will be used. Domain knowledge from prior experiences can help us to determine probable scenarios and associated risks. It can, however, also be misleading. Domains change over time and every organization does business in its own way. Therefore we should know about the business rather than about the domain.**

Does the testing of software differ from one domain to another? And if so, how important is domain knowledge in dealing with those differences? In this article, I will investigate the validity of the assumption that domain knowledge is an advantage for a software tester.

### *Software as a tool*

In some cases software is the product. This is true for games. The product is a collection of files that delivers value directly to the user. In other cases software is part of the product. This is true for mobile telephones, navigation systems and pretty much every other electrical device that is produced nowadays. Often, however, software is not part of the product, but it is only there to support the delivery of a service. This is true for information systems in banks, insurance companies and public organizations. Every thinkable organization uses software to support its business. This kind of software delivers its value directly to the professionals and only indirectly to the customers. This kind of software is a tool.

In the remainder of the article, I will talk about testing software that is used as a tool. It is this kind of software that is used by professionals in different domains.

### *Domain knowledge*

It helps to have experience with specific applications. You know the menu structure. Perhaps you've made a few mistakes in the past from which you have learned. In other words, you are past the steep bit of the learning curve. Some of these applications are domain specific, which means that domain knowledge is built into them. However, having knowledge of such an application is not the same as having knowledge of the domain.

Domain knowledge is knowledge about facts and rules in a distinct line of business. To what extent do we need domain knowledge to test software?

### *Testing and checking*

Regardless of the testing methodology, there are two basic styles of software testing, confirmative and non-confirmative. The goal of confirmative tests is to show that software does work as it is supposed to. The goal of non-confirmative tests is to show that software does not work as it is supposed to. These are very different goals that require different means.

Confirmative testing requires documentation of some sort. This documentation tells us how the software should work under specific circumstances with specific inputs. We check this. This testing style is often done in a scripted way. Since confirmative testing is based on specifications, it should be done objectively. Prior experience in the same domain is more likely a hindrance than an advantage.

Confirmative testing is done in the earlier stages of software development. Once the software does more or less what it is supposed to do, testing can take a more non-confirmative character.

Non-confirmative testing requires imagination. We imagine what could happen and observe the way in which software reacts to different inputs and circumstances. Then we judge whether this reaction is adequate. In order to do this, we need an understanding of the way in which the software is used. This testing style is often done in an exploratory way. In order to be effective at non-confirmative testing which involves thinking up and executing plausible scenarios and correctly judging the software's reaction, some domain knowledge is needed.

This knowledge can be acquired by talking to business representatives from the organization. It is dangerous to rely on knowledge acquired in previous assignments.

### *Facts and rules*

Let's suppose you have worked as a tester for an insurance company. During this assignment you have acquired some knowledge about the insurance domain. Your next assignment is in another

insurance company. Because of your domain knowledge you were the one that got the job. After a while you realize that a lot of things you knew about the old insurance company don't apply to the new one.

An insurance company offers numerous different insurances with different conditions. Someone working for the insurance company should know as much as possible about all the different insurances, especially in a role where this knowledge is needed in communication with clients or colleagues. When testing the software used in this insurance company, it might be handy to have extensive knowledge about all the different insurances. However, since you're not in direct contact with clients or insurance professionals, it is probably enough that you know where to find the descriptions of the different insurances and their conditions. Knowledge about the specifics of insurance policies will not help you if you need to test software in another insurance company with different insurances.

Before testing software you should explore the requirements, and not only the ones that are written down. You should try to get to know your user group. Although there are similarities between companies in the same domain, there are also differences. Even within companies there are differences between departments. Specific knowledge, based on user groups you have encountered before, might cloud your vision.

The facts and rules of a domain are not fixed. At a specific level they vary between, and even within, organizations. At a more general level they may change over time. Regulations, and even laws, are not written for eternity. Therefore you need to verify domain knowledge you have acquired in previous assignments. In doing so, you just might learn something new.

### *Software quality*

Testing can never be complete. When software is actually used by professionals in their daily work, combinations of circumstances and inputs will occur that have not been tested. In order to test the optimal subset of all possible combinations, we need a genuine interest in the software and its users, some imagination and, of course, our testing skills.

Logically, we can never reach the level of domain knowledge that the actual users of the software have. This is because we have our own domain, which is software quality. Software quality manifests itself in the interaction between users and software. Knowledge about the use of software is part of our domain. We need to know what users do and how they do it.

As software testers we operate in between business and technology. To do our job well we have to know them both to some extent. We need to know more about the technology than business people and more about the business than technical people. There is only one sort of domain knowledge that really makes us better at our job and that is knowledge of the software testing domain.

## > biography

**Bert Wijgers**
*is a test consultant with Squerist, a service company in the Netherlands. Squerist focuses on software quality assurance and process optimization. Its mission is to inspire confidence through innovation.*

*Bert holds a university degree in experimental psychology for which he did research in the areas of human-computer interaction and ergonomic aspects of the workspace. He has worked as a teacher and trainer in different settings before he started an international company in web hosting and design for which a web generator was developed. There he got the first taste of testing and came to understand the importance of software quality assurance. Bert has a special interest for the social aspects of software development.*
*He uses psychological and business perspectives to complement his testing expertise.*

*In recent years Bert has worked for Squerist in financial and public organizations as a software tester, coordinator and consultant. He is a regular contributor to Testing Experience.*



# Using realistic user behavior simulation to detect e-commerce performance problems

*by Christian Obst & Stefan Lehmann*

In informatics, the word "performance" describes the ability of a data processing system to work on a task in a defined way. The speed of an e-commerce web application can be more easily expressed in terms of response time. The response time of a web application depends on many factors such as the network used, the type of browser and the hardware configuration of the server, on which the application is hosted. So it is quite possible that the same client using different network bandwidths gets different response times. It also means that one request could claim different durations in different browsers on the same computer.

$$t_{Response} = t_{Browser} + t_{Netz} + t_{Server}$$

How "fast" the application is depends on the human perception. As far as the time is concerned, the response time can be subdivided into three sections. For response times up to 0.1 seconds, one feels that the application reacts immediately. For response times up to 1.0 seconds, the user finds the processes trouble-free and can work continuously. For times up to ten seconds, most users remain concentrated and pursue the course of the software (Miller 1968). However, it is not always true that faster is better. Quick answer times can lead to more mistakes in the working process. The optimal response time depends on the nature of the request. Response times up to about two seconds are adequate for most e-commerce tasks. What are the consequences of exceeding these recommended limits? A survey by Computerworld in 1986 (Rushinek 1986) reveals that speed is one of the most crucial factors for user satisfaction. 15,288 subscribers elected speed to be the top criteria for user satisfaction. Three of the top five answers were performance related. In order to judge, two e-commerce user groups are regarded: On the one side are the dealers, who cannot be as productive if they must wait for the system. On the other side are the customers, who want to have information about products and prices. Since on an e-commerce system several users interact at the same time, it is very important to know how many users act in parallel. In order to make a statement about the requirements for such a system, it is necessary to know how many requests per second are to be processed by the system. The test scenario was provided on the basis of real e-commerce data to arrange the execution of the performance tests realistically. The goal was to find out how many requests a shop system expects (request per shop per hour).

To achieve that it is necessary to perform an analysis of the system load. Log file data of five providers over a period of 25 weeks were analyzed [Lehmann 2010]. An average weekly pattern in which an average course of the day can be seen, is defined by

$$N_{t_{d,h}} = \frac{\sum_{W=1}^{25} n_{t_{d,h},W}}{25}$$

Relative to the sums of the times of day on the sum of requests each day, you get a percentage progression.

$$N_{t_{d,h_r}} = \frac{N_{t_{d,h}}}{N_d}$$

In order to later make a statement about the daily load of a single store, the average total number of requests per store per day must be recorded.

$$N_d = \sum_{h=0}^{24} N_{t_{d,h}}$$

Because the course of each day is very similar, it affords the arithmetic average over all days.

$$N_{t_{d,h_{adv}}} = \frac{\sum_{d=1}^{7} N_{t_{d,h}}}{7}$$

For better statistical significance, the statistics of the 5 providers were equivalent averaged (Table 1).

At this point, relative distribution is used because absolute distribution of the individual provider due to the size of installations

# Probador Certificado Nivel Básico

## Tester profesional de Software

Formación para el Probador Certificado - Nivel Básico
de acuerdo al programa de estudios del ISTQBâ

República Argentina


### Docente: Sergio Emanuel Cusmai

**Co - Founder and QA Manager** en QAustral S.A.
**Gte. Gral.** de Nimbuzz Argentina S.A.
**Docente** en diplomatura de Testing de Software de UTN - 2009.
**Titular y Creador** de la Diplomatura en Testing de Software de la UES XXI - 2007 y 2008.
( Primer diplomatura de testing avalada por el ministerio de Educación de Argentina).
**Team Leader** en Lastminute.com de Reino Unido en 2004/2006.
**Premio** a la mejor performance en Lastminute.com 2004.
**Foundation** Certificate in Software Testing by BCS - ISTQB. London – UK.
**Nasper** - Harvard Business school. Delhi – India.

**Casa Central:** + 54 0351 4817951 - **Tel Bs As Argentina:** 011 53685847 - **Tel Londres UK:** 00442033183814
**Tel San Francisco.** EEUU: 001 14157041515 - **E-mail: info@qaustral.com**

| | Tue 14:00-15:00 | Tue 15:00-16:00 | Tue 16:00-17:00 | Tue 17:00-18:00 | Tue 18:00-19:00 |
|---|---|---|---|---|---|
| **Provider 1** | 7,59 | 8,51 | 7,90 | 8,00 | 6,04 |
| **Provider 2** | 6,58 | 7,83 | 8,16 | 8,33 | 7,57 |
| **Provider 3** | 4,54 | 5,04 | 7,29 | 8,96 | 8,99 |
| **Provider 4** | 8,44 | 8,70 | 8,14 | 4,94 | 3,86 |
| **Provider 5** | 7,59 | 8,51 | 7,90 | 8,00 | 6,04 |
| **Average** | 6,95 | 7,72 | 7,88 | 7,65 | 6,50 |

Table 1: Summary of all providers (BE)

| | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Shops |
|---|---|---|---|---|---|---|---|---|
| **Provider 1** | 135 | 138 | 133 | 132 | 130 | 118 | 133 | 23981 |
| **Provider 2** | 194 | 194 | 191 | 188 | 180 | 165 | 175 | 2200 |
| **Provider 3** | 171 | 172 | 166 | 161 | 151 | 131 | 148 | 1350 |
| **Provider 4** | 152 | 151 | 144 | 142 | 143 | 133 | 139 | 294 |
| **Provider 5** | 87 | 87 | 87 | 92 | 84 | 80 | 80 | 1520 |
| **Average** | 148 | 148 | 144 | 143 | 138 | 125 | 135 | |
| **Per Hour** | 7 | 7 | 7 | 6 | 6 | 6 | 6 | |
| **Total Per Day** | **140** | | | | | | | |
| **Total Per Hour** | **6** | | | | | | | |

Table 2: Result of the analysis

varies widely. From this information it is now possible to see the hour at which most requests are generated.

Since there are different types of request, the web server logs were analyzed for the number of different requests to obtain a relative distribution. In this evaluation back-end (BE) and front-end (FE) were considered separately. Based on this distribution, the test scripts can later be scaled. The outcome of this is that for the BE and the FE each of the 13 different request types arise. On average, a shop receives per day 21 BE and 140 FE requests (Table 2). Both for the BE and the FE, the time when most requests arrive is between 16:00 and 17:00 hours. This corresponds to 7.75% (BE) or 7.07% (FE ) of the daily incoming requests to the system. These data are the minimum requirements for performance test cases to simulate a realistic e-commerce system.

Due to the fact that the ePages e-commerce solution is rented to providers with a minimum license for 500 shops, the load is designed to that minimum shop count.

$$N_{BE} = 21 \times 0.0775 \times 500 = 814$$

$$N_{FE} = 140 \times 0.0707 \times 500 = 4949$$

A test should be used during the development lifecycle to verify the quality. One performance measurement was set to run in 15 minutes. Therefore the numbers of requests have to be reduced to one quarter:

$$N_{BE25\%} = N_{BE} \times 25\% = 204$$

$$N_{FE25\%} = N_{FE} \times 25\% = 1238$$

The analysis of the user behavior results in figures 1 and 2. Multiplication of the probability of the itemized request types by number of 500 shops expected requests results in the sum of the request that have to be covered by the test scripts.

To evaluate the results of the measurements in an easier way, a quality criterion is required. Basically it is a school grade system that reflects the measured response times. The general target value is a response time of 0.5 seconds, which would give a 1. Every response time that is greater than or equal to 3 seconds is a mark 6. To calculate the quality index, the average response times of the itemized requests have to be divided by the target value and multiplied by their probability of occurrence. The sum of the values for all requests results in the quality.

$$Q = \sum \frac{t_{request}}{t_{target}} \times p(request)$$

Due to the relation between the number of FE and BE requests being about 6:1, it is reasonable to evaluate both parts separately from each other. Otherwise the results for the FE requests would falsify the result for the whole system.

After identifying which system load can be simulated, the next step must be creation of the appropriate scripts. For this, in a subsequent analysis, the use of Selenium (http://seleniumhq.org) and JMeter (http://jakarta.apache.org/jmeter) are evaluated. The use of Selenium has its advantage in that the use of a "real" browser can be simulated. If the system load is simulated by JMeter scripts the test requirement of the test client system can be reduced and integration into an existing unit test environment can easily be done. Whether the two tools can be used to simulate the system load and whether the resulting performance data of the e-commerce application can be derived, will be examined in a next step.

A. and S. Rushinek: What Makes Users Happy? in: Communication of the ACM, 29, vol. 7 july 1986

Lehmann, Stefan: „Konzeption, Umsetzung und Analyse von Performancemessungen eines E-Commerce Systems zur Beurteilung von Server-Hardwarekonfigurationen und zur Performancesicherung im Software-Entwicklungsprozess“, University of Applied Science, Jena, 2010

Miller, Robert B.: Response Time in M-Computer Conversational Transactions in: Proceedings Spring Joint Computer Conference, Montvale 1968, pp. 267-277

## > biography

**Stefan Lehmann**
*finished his studies in 2010 and now works at ePages GmbH as technical tester. He is responsible for the implementation of performance measurement and development of test automation.*

**Christian Obst**
*works at ePages GmbH as Quality Assurance Manager. Christian has more than 10 years of experience in software QA.*



# Pragmatic test approach as key to unlock infrastructure's doors

*by Wim Demey*

*Nowadays most test professionals are involved in the domain of application development. A lot of test methodologies, templates and tools help them to do their daily work in a confident and efficient way. But what about testing in the domain of infrastructure? Why is this domain for most of us a real black and secret box? Which arguments can we bring in the discussion to prove the need and added value of structured testing? And if we succeed to open the door, which strategy and methodology can we apply? It is clear that infrastructure testing raise a lot of questions which will be answered in this article.*

*Several years ago I was accidentally involved in an infrastructure project. It was about a company-wide upgrade project of 12,000 clients. Since that project I have extended my experience and knowledge in other infrastructure-related projects like workstation maintenance, database moves and set-up of Citrix foundation. As this domain was completely new for me, I was surprised by the lack of useful information about this topic and the absence of test professionals in this domain.*

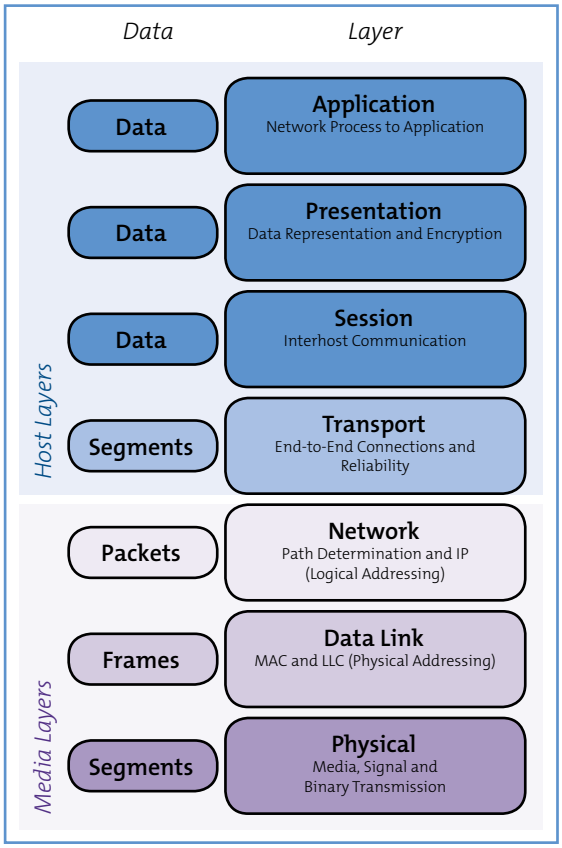



Figure 1: OSI model

## What is infrastructure testing?

To counter the first shortcoming, I searched for a definition of infrastructure testing. However, after some attempts it seems that amongst the long list of internationally widely accepted testing definitions there is not yet a good one for infrastructure testing. So we have to formulate our own definition which is not easy because it depends on your particular perspective.

### 1. OSI model perspective

The OSI model (ISO reference for Open Systems Interconnection model) represents a big part of the activities which take place in infrastructure.

The benefit of this model is a clear and precise scope. So you can define infrastructure testing as any action related to one of the OSI layers.

There is, however, a disadvantage in using this narrowed view. It implies that we will miss some interesting areas for testing like management of workstations, deployment of default software for all end users, set-up of database, web foundations.

### 2. White/black box perspective

In some way everything about infrastructure could be considered as a black box for testers. They don't need to care about it just as they don't care about unit testing which is the responsibility of the developers. In this context, infrastructure testing is just a type of white box testing and doesn't need an own definition. However, this is not completely true as white box testing is oriented on application development whereas infrastructure activities are mainly oriented on the installation and/or configuration.

### 3. Organization structure perspective

Most companies have split up their IT department into several sub-divisions of which infrastructure is one. This division covers not only hardware but also software related infrastructure activities (e.g. management of databases, servers, set-up of LANs,

upgrading and patching of operating system) and the processes around it.

In fact, the infrastructure team delivers a complete and reliable infrastructure platform for applications to internal/external customers.

With these perspectives in mind, I have tried to formulate a definition. Any ideas for alternatives, suggestions are welcome!

> Infrastructure testing is any test activity performed as result of an intervention made on hardware, network and/or software components which are part of an integral infrastructure platform in order to facilitate applications in a managed and controlled way.

## Why are we not involved?

Since the trend towards cloud computing and virtualization, we cannot ignore the increased importance of a reliable infrastructure. When using the cloud this more or less means that the cloud becomes part of your IT infrastructure. Gartner's report[1], in which system infrastructure is described as one of the six cloud computing layers, confirms this statement as follows: "System infrastructure services are the most basic and fundamental form of cloud computing service, and parallel the infrastructure and data center initiatives in IT."

From a quality assurance point of view, it is obvious that testing must and will play a crucial role in the infrastructure area. This assumption gains even more credibility as Gartner[2] expects that cloud application infrastructure technologies need seven years to mature divided over 3 transition phases of which the last ends in 2015.

Paradoxically the doors of infrastructure departments remain often closed for test professionals. Why? From both sides arguments are brought into the discussion about the role and need of testing.

- No development

Infrastructure engineers argue that they don't develop software. They only configure systems and components which are in most cases third-party packages (e.g. MS SQL server, Exchange, Symantec Antivirus). Moreover they try to avoid as much as possible any additional customized developments to interface with other applications.

- Thorough testing is the vendor's responsibility

Going further on from the previous argument, it seems obvious that vendors of those packages or hardware components do thorough testing. They need to ensure that the functionalities of their products behave as expected.

In this mindset there is not much additional testing needed. A quick sanity test should be sufficient as kind of acceptance of the package/component.

- No resources available

Teams of infrastructure engineers are often not equipped with dedicated test resources. Moreover their main task is ensuring that the day-to-day core business operates smoothly without interruptions at any time. So testing is for them more an idle period activity.

- Fear keeps them in the comfort zone

Historically most test professionals started their career in the domain of functional application testing. They were happy sitting behind their application under test (AUT) and mimicking the actions of the end users. A first wave of uncertainty popped up by the introduction of SOA and other GUI-less technologies. People felt less confident about them because they had to look literally behind the screens to verify what they were testing. Digging into xml-files, firewall logs, group policies, database traces... is still for many testers not their cup of tea.

- We are not like them

Sometimes project management offices (PMO) require that infrastructure project leaders apply the same project & test methodology as their colleagues of application development. This copy & paste method doesn't work out though. People are overwhelmed by methodology overhead and only see the disadvantages of using a structured and phased approach. PMO's have to understand that infrastructure projects are unique with their own peculiarities and approach.

## Quality is the key

Although this paradox exists, one thing must be crystal-clear for all parties involved in infrastructure projects: testing is necessary. The key to unlock the door is quality assurance which is the same rationale for testing in non-infrastructure projects. And when you are interwoven with the cloud, your infrastructure depends on services of which you don't know the level of quality. So testing is one of the handles to ensure that those external cloud services and applications still meet the business and operational requirements you have.

This fits also perfectly in the ITIL Continual Service Improvement (CSI) mindset, which is a hot topic in infrastructure today. The process objective of CSI is to use methods from quality management – of which testing is an activity - in order to learn from past successes and failures. This means that the effectiveness and efficiency of IT services and processes are continually improved.

The following example illustrates that quality is not just a hollow concept in infrastructure projects. The workplace upgrade project delivers a brand new image with new versions of default software (e.g. Office, Acrobat Reader, Internet Explorer, SAP GUI, ...) to every workstation. Technically this was not too complicated as the image consisted of a bundle of packages and was pushed to every workstation which had been completely scratched. So one might wonder if and to which degree huge test effort was needed? Due to the wide impact of this project, any issue or deviation would have been directly noticed. Moreover some people were very sceptical and conservative for these changes. The project leader understood that poor quality would definitely kill a smooth roll-out of the image. Hence we concluded that all current applications on the radar of the end users must be regression tested against the new workplace image.

## Pragmatic approach to build trust and prove value

Now time has come for test professionals to leave their comfort zone and think about a practical test methodology and approach

1 *Gartner RAS Core Research Note G00170572, Ben Pring, Claudio Da Rold, 14 September 2009, RA8, 07042010*

2 *Garter, http://www.gartner.com/it/page.jsp?id=871113, 2 February 2009*

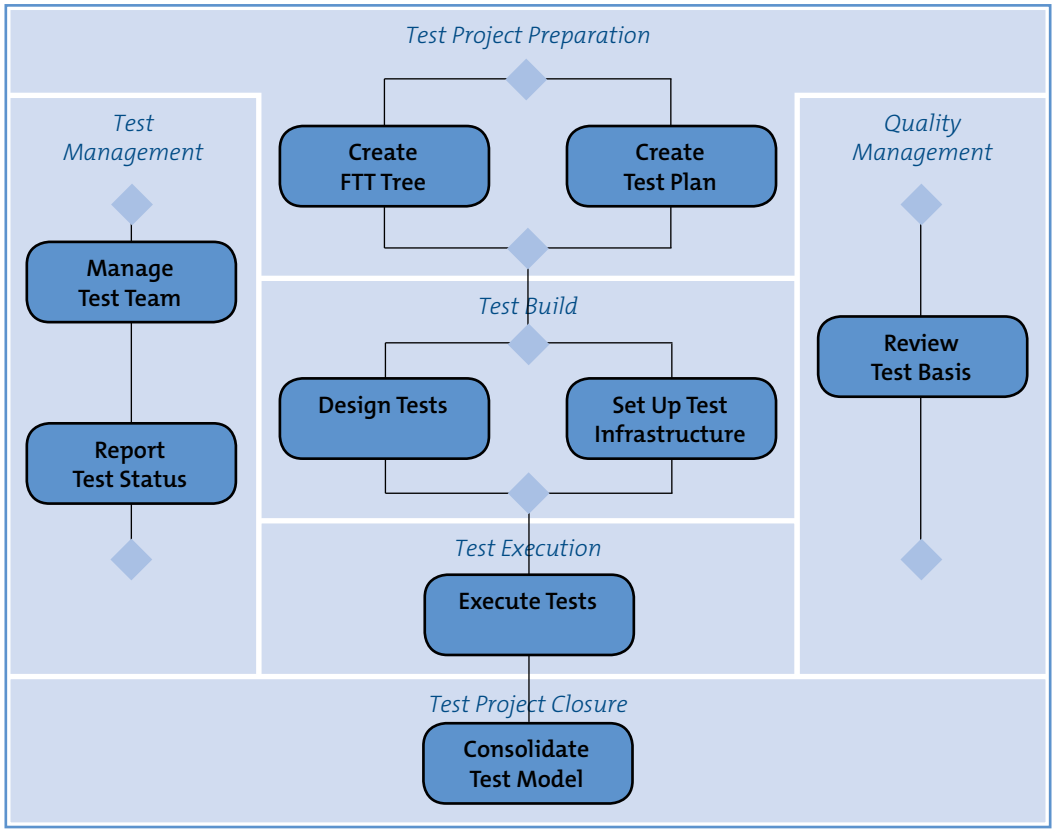



Figure 2: STBox Essentials workflow

for infrastructure testing. The good news is that a structured and phased approach –like you normally apply in application testing - is here valid as well.

However, bear the following caveats in mind if you want to build trust and prove the added value of testing in infrastructure projects.

### 1. Agile character of project activities

Configuration activities (e.g. enable/disable settings, define group policies, open a port on a fire wall) are quicker executed than coding of a functionality.

This means that the timeframe for test design and test execution is very short. Most of the time there are several iterations of configuring and testing until the expected result is achieved. In fact, this way of working seems very agile and requires great flexibility and quickness of reaction of testers.

### 2. Procedural context

Larger organizations where infrastructure is a top priority for their daily operations have often a strong procedural approach. They want to minimize the risk of downtimes by setting up and applying strict procedures which describe what to do if a component must be upgraded or replaced. To some extent it could be interesting to approach these procedures as if they where code:

- Code that needs to ensure the requirements are met (e.g. company rules about availability/responsiveness of IT infrastructure).
- The code needs to be tested.

If you have a procedure to replace a workstation, does this actually lead to an operational PC in the end?

- Code needs to be maintained.

Are the procedures updated when a new printer has arrived?

### 3. Content of project phases depends of project type

Although the sequence of phases is the same in each infrastructure project, the content and activities depend on the type of project. The figure below illustrates that regression testing takes place in different project phases.

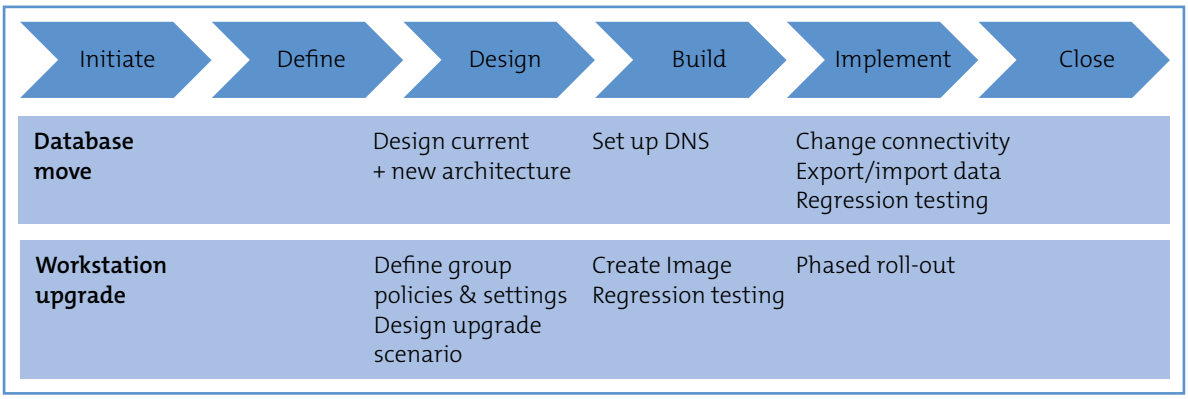



Figure 3: Different content of project phases depending on type

### 4. Careful selection of test types and techniques

As we do not have plenty time and money, risk-based testing in infrastructure projects plays a crucial role. The test effort must be in proportion to the goals, and therefore the appropriate test types and techniques are defined in the test strategy.

Here again, the choice depends on the type of infrastructure project as shown in the table.

Of all techniques, the CRUD gives you a jump start, although the content of each step is slightly adapted to an infrastructure project. In this context I'm supporting the definition of Olivier Brian[3] who uses a more appropriate definition: ICUR (Install, Configure, Update and Remove).

| Test Type | Work-station upgrade | Database move | Citrix foun-dation |
|---|---|---|---|
| Unit test | - | - | - |
| Integration test | - | x | - |
| Regression Test | x | x | x |
| Security test | - | x | x |
| Portability test | x | - | - |
| Performance test | - | - | x |

Figure 4: Test types

### 5. Lean and mean test management

It is clear that infrastructure projects require a structured but adapted test process. When you try to implement a full-blown test methodology out of the box, it won't work. The key of a successful test approach lies in using simple templates, short communication lines and producing deliverables which have a real added value for everybody. Unlike application development projects, the role of test manager is broader and you must not be scared of doing some hands-on stuff as well. In my projects I have experienced that this way works and helps to clean up objections of infrastructure colleagues who still have their doubts about the added value of testing.

### 6. Defect analysis, liaison between applications & infrastructure

Defect tracking and especially the analysis is different from application projects. Sometimes issues - due to improper configuration of settings - could remain hidden for a while (e.g. a roaming profile which becomes corrupted after several log-ons). At that moment the root cause is not always clear. Moreover when the issue impacts the application(s) running on top of the infrastructure platform, the fight about responsibility could start.

In some cases an application is no longer adapted to newer versions (e.g. VBA code developed in Office2000 which is not compatible with Office2010). From an infrastructure point of view, the application team has to adapt their application while they argue that they haven't asked for an upgrade to the new Office version.

As test manager you have to mediate in such discussions and look with both parties for a feasible solution taking the costs of each possibility into account.

## Conclusion

The domain of the infrastructure contains a lot of opportunities and challenges which test professionals will face more and more as cloud computing becomes still more mature. There are a lot of things to explore and our knowledge about testing these rich infrastructure features still needs to be elaborated. With a structured but light-weight and integrated test approach you can build up confidence and proof for the real added value of testing. After a while infrastructure engineers will even be happy to get aid and support for their testing activities. So come on, leave your comfort zone and jump into this new world of testing!

### > biography

**Wim Demey**
*is a Senior Test Consultant working for CTG Belgium. He started his career in 1997 and has extended his knowledge and expertise over the years by working in several areas (pharma, utilities, telecom, non-profit, social security) and roles (test manager, test tools specialist, coach, presales).*
*Driven by an eagerness to learn new things he is always looking, where he can cross the borders of traditional testing with a special interest for technical topics (infrastructure testing, test tools). This allows him to get not only a good high-level view of a project, but also being able to discuss with customers and project teams about the low level details.*
*His motto is "Make IT happen", which means doing your daily job in a pragmatic, constructive and collaborative way, without being afraid of the real hands-on stuff as well. That's the way to deliver the real added value that customers expect from a consultant.*
*Over the years he obtained several certificates and was regularly a speaker at national and international conferences and seminars.*
*For more information, questions or remarks, you can contact him by mail: wim.demey@ctg.com*

3 Brian Olivier, http://www.infratester.eu/2009/02/icur-infrastructure-crud.html




# V&V of medical device software

*by Nadica Hrgarek*

Medical device software differs from hardware and is characterized by a high complexity due to a rapid rise in features, by an increased attention to human factors and usability engineering requirements, by constantly changing requirements, by ever faster time-to-market (which requires faster product development), by a requirement that the software does not wear out, by systematic software failures (i.e. probability that the failure will occur must be assumed 100%), by the fact that most software errors are traceable to requirements and design, and by the risks posed by software which is easy to change but may have a significant impact on the whole software system. Dealing with these challenges requires effective and efficient project management, excellent engineering practices by the development teams, and controlled processes to be used for the safe *design*, *development* and *maintenance* of medical device software.

Driven by the need to protect patients, users and third parties, the regulators are very interested in medical device software. Adequate documentation, taking into account the principles of the software development life cycle, software verification and validation (V&V) activities and their documented outputs, change control processes, managing SOUP (Software Of Unknown Provenance) items, well integrated software risk management and a usability engineering process, and the full *traceability* throughout the software development life cycle, all play an important role for medical device manufacturers when facing the regulators.

The goal of this article is to provide an overview of the verification and validation regulatory requirements, activities, and methods related to medical device software.

The **IEEE 610.12-1990** standard defines *verification and validation* as *"the process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements"*.

### *Regulatory requirements on medical device software*

According to the amended European Medical Device Directive (MDD; **93/42/EEC**), the following fall under the classification of *medical device software*: a) software which is a *medical device itself* (i.e. stand-alone software is now defined as an active medical device) or an *accessory to a medical device*; and b) software which is a *component and integral part of a medical device*. Software for general purposes which is not covered by the MDD may include, but is not limited to: software used for administrative purposes (e.g. handling of patient files and data, laboratory information systems), software used for the education of medical doctors, software used as a tool within the overall design and manufacturing processes of the medical devices (e.g. compilers, configuration management systems, ERP, inventory control systems, SPC tools, etc.).

Operating within a highly regulated industry, medical device manufacturers must comply with all regulatory requirements which are applicable on the market where the product is intended to be sold. For example, in order to get your medical device product on the U.S. market, you need to comply with the federal laws and regulations governing medical devices in the U.S. The basic regulatory requirements for medical devices distributed in the U.S. are: registering your establishment with the FDA (Food and Drug Administration) and registering a U.S. Agent for any foreign establishment (**21 CFR Part 807**), medical device listing (**21 CFR Part 807**), submitting any required premarket notification 510(k) (**21 CFR Part 807 Subpart E**) or premarket approval (PMA) (**21 CFR Part 814**), investigational device exemption (IDE) for clinical studies (**21 CFR Part 812**), quality system regulation (QSR) / good manufacturing practices (GMP) (**21 CFR Part 820**), labeling requirements (21 CFR Part 801), and medical device reporting (MDR) (21 CFR Part 803). The FDA published several guidance documents related to medical device software (refer to Table 1). For example, the FDA's **Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices** determines the type and extent of software documentation required for premarket submission which depends on the *Level of Concern* (i.e. *Major*: death or serious injury of the patient or operator is possible, *Moderate*: minor injury is possible, or *Minor*: no injury to the patient or operator is possible). The FDA looks very thoroughly at software verification and validation, and at software changes during the inspection of the manufacturer's quality management system.

Medical device software is also subject to many standards and regulations. **IEC 62304** is a harmonized standard that specifies pro-

cesses associated with the software development life cycle (refer to Figure 1) and a software safety classification scheme (i.e. *Class A* – lowest risk: no injury or damage to health is possible, *Class B* – medium risk: non-serious injury is possible, or *Class C* – highest risk: death or serious injury is possible) aimed at improving the quality and safety of medical device software. Safety classification of a medical device software system and the corresponding software items is based on the severity of the *hazard* resulting from failure of the software, assuming that the failure will occur. The standard allows different software items of the same software system having different safety classifications. It was derived from the approach and concepts of the ISO/IEC 12207:1995 standard, which defines requirements for software life cycle processes in general (i.e. not restricted to medical device software). IEC 62304 is applicable to *development* and *maintenance* of *medical device software* (e.g. release of a new software version, new intended use, etc.), when the software is itself a medical device or when software is embedded or integral part of the final medical device. Since March 2006, this standard is also recognized by the FDA and replaces U.S. standard ANSI/AAMI SW68:2001. IEC 62304 does not prescribe an organizational structure, a specific software development life cycle model (e.g. V-model, waterfall, spiral model, etc.), or the content and format of the documentation to be produced. Major principles which promote safety for medical device software are: quality management system, risk management process and software engineering. IEC 62304 assumes that medical device software is developed and maintained within a quality management system and a risk management system.

Regulatory requirements for quality management systems of medical devices are well addressed by the international standard **ISO 13485** (and by **21 CFR Part 820** in the U.S.), whereas the risk management process is addressed by the international standard **ISO 14971**. The risk analysis early in the software development life cycle ensures that the safety features are considered and built into the design.

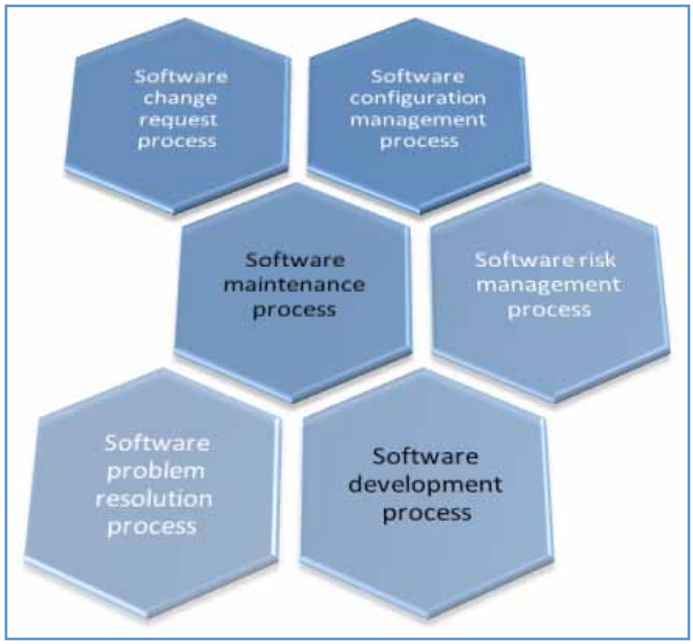



Figure 1: IEC 62304 software life cycle processes for medical device software

The **IEC/TR 80002-1** technical report provides guidance for implementing a safety risk management process for medical devices containing software, as part of the overall risk management process. This technical report attempts to explain how major sections of the ISO 14971 standard relate to medical device software with reference to the IEC 62304 standard.

The **IEC 60601-1** standard can be used for PEMS (Programmable Electrical Medical Systems) validation. Clause 14 of the IEC 60601-1 standard (formerly the collateral standard IEC 60601-1-4:1996 + A1:1999) incorporates the following requirements for PEMS: documentation requirements, risk management plan, documented PEMS development life cycle, problem resolution system, risk management process, requirements specification, architecture, design and implementation, verification, validation, modification/change control, and connection of PEMS by network/data coupling to other equipment.

Poor interface design and usability of software functions that involve user (i.e. patient, operator, third party) interaction with the medical device increase the probability of medical errors. Since usability relates to *safety* (i.e. freedom from unacceptable risk), it needs to be considered as integrated part of the design and development process of the medical device. **IEC 62366** is a safety-related harmonized standard which specifies a usability engineering process that a medical device manufacturer can use to analyze, specify, design, verify and validate the usability of a medical device. The usability engineering process assesses and mitigates risks caused by usability problems associated with correct use and use errors which may occur in normal use. All results of the usability engineering process shall be documented in a Usability Engineering File. As provided in the standard, usability is limited to characteristics of the user interface. The user interface includes all means by which the user and the medical device interact (refer to Figure 2), and this interaction is examined using the *usability tests*. Usability verification and validation activities should be performed iteratively throughout the design and development process using the quantitative and/or qualitative methods and techniques (e.g. *cognitive walkthroughs, design audits, prototyping, usability tests, questionnaires and surveys, interviews, risk analysis, use error analysis, expert reviews, participatory design, simulated clinical environments, workload assessment*, etc.) before use on actual patients.

The **IEC 60601-1-6** collateral standard specifies a usability engineering process for medical electrical equipment. It has been aligned with the usability engineering process in the IEC 62366 standard.

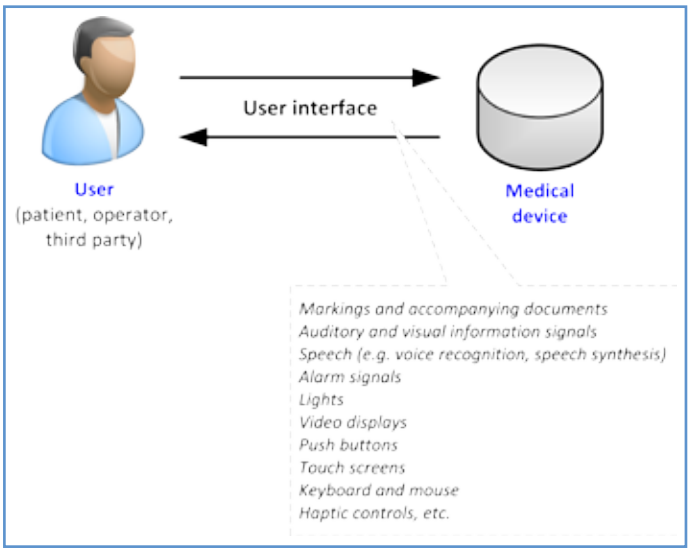



Figure 2: User interface according to IEC 62366 and IEC 60601-1-6 standards

Table 1: Standards, regulations and guidelines for medical device software

| ID | Title |
|---|---|
| ISO 13485:2003 | Medical devices – Quality management systems – Requirements for regulatory purposes |
| 21 CFR Part 820 | Code of Federal Regulations Title 21 Part 820 Quality System Regulation |
| IEC 62304:2006 | Medical device software – Software life cycle processes |
| IEC 62366:2007 | Medical devices – Application of usability engineering to medical devices |
| IEC 60601-1-6:2010 | Medical electrical equipment – Part 1 – 6 General requirements for basic safety and essential performance – Collateral standard: Usability |
| ISO 14971:2007 | Medical devices – Application of risk management to medical devices |
| IEC/TR 80002-1:2009 | Medical device software – Part 1: Guidance on the application of ISO 14971 to medical device software |
| IEC 60601-1:2005 | Medical electrical equipment – Part1: General requirement for basic safety and essential performance, Part 14 Programmable Electrical Medical Systems (PEMS) |
| - | Guidance for Industry and FDA Staff; Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices |
| - | General Principles of Software Validation; Final Guidance for Industry and FDA Staff |
| - | Guidance for Industry; FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices |
| - | Guidance for Industry; Cybersecurity for Networked Medical Devices Containing Off-the-Shelf (OTS) Software |

### *Verification*

The **IEEE 610.12-1990** standard defines *verification* as: *"(1) the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase; (2) formal proof of program correctness"*.

According to the FDA's **General Principles of Software Validation** guidance document, *"software verification provides objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase"*.

The **IEC 62304** standard requires verification of medical device software at several software development stages. IEC 62304 determines the following software development sub-processes: software development planning, software requirements analysis, software architectural design, software detailed design, software unit implementation and verification, software integration and integration testing, software system testing, and software release. The verification activities shall be carefully planned and documented. The documentation and the software verification efforts are scaled according to the criticality of medical device software. The following outputs/sub-processes of the software development and maintenance processes shall be verified: software requirements [Class A, B, C], software architecture [Class B, C], software detailed design [Class C], software units [Class B, C], software integration [Class B, C], software integration test procedures [Class B, C], software system testing [Class B, C], risk control measures [Class B, C], software changes [Class A, B, C], and software problem resolutions [Class A, B, C]. In order to satisfy these verification requirements, the following verification methods and techniques may be used: *formal reviews* (e.g. *design, documentation, code reviews), traceability analysis, walkthroughs, audits, manual and automated software tests (i.e. unit, integration, system, regression tests*), etc.

According to Clause 14 of the **IEC 60601-1** standard, verification of PEMS is required for all functions that implement basic safety, *essential performance* (i.e. the performance needed to achieve freedom from unacceptable risk) or risk control measures, and shall be performed according to the Verification Plan. Clause 14 also provides some examples of verification methods and techniques: *walkthroughs, inspections, static analysis, dynamic analysis, white box testing, black box testing, and statistical testing*.

*Design reviews* evaluate existing or proposed design to detect and resolve issues and potential problems with the design. These reviews shall be planned, conducted at appropriate stages of the medical device's design and development process, and documented. Formal *documentation reviews* can be used to find errors and improve the quality of technical documents under review (e.g. Software Requirements Specifications (SRS), Software Design Specifications (SDS), Software Architectural Design, Test Cases, Test Plans, Risk Analysis, etc.). *Code reviews* verify that the coding guidelines have been followed and that the design has been appropriately implemented. Code reviews also increase the number of employees who understand the code. A *source code traceability analysis* verifies that code is linked to established specifications and test procedures. Conducted by trained internal auditors, *process audits* are helpful to verify if the inputs, activities and outputs of the software development and maintenance processes are in accordance with the specified requirements. *Product audits* verify if a particular medical device software product conforms to product specifications (e.g. software requirements specifications, use cases, software design specifications, usability specifications, etc.), applicable standards and regulations, and customer requirements. *Unit tests* are based on a detailed knowledge of the source code and the software's architectural and detailed design in order to detect and correct errors. Whenever possible, these tests should be mainly executed automatically. The goal of *integration testing* is to test integrated software items to detect and correct errors arising from the combination of software units and external hardware. *System testing* is testing of the complete system prior to validation. IEC 62304 allows combining integration testing and software system testing. As required by IEC 62304, all anomalies found during software integration, integration testing, system testing as well as testing, retesting or regression testing of software items and systems following a change, shall be documented. Results of *software regression testing* conducted to make sure that features that already worked are not affected by the change, shall be documented. *Statistical software testing* is an advanced verification technique used to test complex software for *reliability*. This type of testing uses randomly generated test data from defined distributions based on an operational profile in order to cover particular areas of concerns.

iseb Accredited Training Organisation

1. akkreditiertes Unternehmen im deutschsprachigen Raum

# ISEB Intermediate (deutsch)

Der ISEB Intermediate Kurs ist das Bindeglied zwischen dem ISTQB Certified Tester Foundation Level und dem Advanced Level. Er erweitert die Inhalte des Foundation Levels, ohne dass man sich bereits für eine Spezialisierung - Test Management, technisches Testen oder funktionales Testen - entscheiden muss. In drei Tagen werden Reviews, risikobasiertes Testen, Test Management und Testanalyse vertieft; zahlreiche Übungsbeispiele erlauben die direkte Anwendung des Gelernten.

Eine einstündige Prüfung mit ca. 25 szenario-basierten Fragen schließt den Kurs ab. Das „**ISEB Intermediate Certificate in Software Testing**“ erhält man ab 60% korrekter Antworten.

**Voraussetzungen**

Für die Zulassung zur Prüfung zum “Intermediate Certificate in Software Testing“ muss der Teilnehmer die Prüfung zum Certified Tester Foundation Level (ISEB/ISTQB) bestanden haben UND entweder mindestens 18 Monate Erfahrung im Bereich Software Testing ODER den akkreditierten Trainingskurs “ISEB Intermediate” abgeschlossen haben - vorzugsweise alle drei Anforderungen.

**Termine**

| | |
|---|---|
| 05.04.11–07.04.11 | Berlin |
| 08.08.11–10.08.11 | Berlin |
| 21.11.11–23.11.11 | Mödling/Österreich |

**€1600,00**

plus Prüfungsgebühr €200 zzgl. MwSt.

http://training.diazhilterscheid.com

Díaz Hilterscheid

**IEC 62366** and **IEC 60601-1-6** require verification of the user interface implementation against the user interface requirements defined in the Usability Specification.

Underestimating the time for software verification activities, changing key features late in development, no code reviews, or incomplete testing due to time pressures, are common challenges related to software verification. To avoid these issues, V&V activities shall be carefully planned and prepared from the beginning of the project

### *Validation*

The **IEEE 610.12-1990** standard defines *validation* as *"the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements"*.

According to the FDA's **General Principles of Software Validation** guidance document, *"software validation is confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled"*.

Validation requirements apply to software which is a component/integral part of a medical device (e.g. blood pressure measuring device, infusion pump, patient cardiac monitor, clinical laboratory instruments, etc.), to software that is itself a medical device (e.g. blood establishment software systems that control donor deferrals and the release of blood products, software for diagnostic image processing, etc.), and to software used in the production of a medical device, or in implementation of the medical device manufacturer's quality management system.

European Directive **2007/47/EC**, which is amends MDD (**93/42/EEC**) and the Active Implantable Medical Device Directive (AIMDD; **90/385/EEC**), added in Annex I a new essential requirement 12.1a for the validation of medical device software: *"For devices which incorporate software or which are medical software in themselves, the software must be validated according to the state of the art taking into account the principles of development lifecycle, risk management, validation and verification."*. This essential requirement is also applicable to SOUP items utilized in a medical device.

The **IEC 62304** standard only covers verification and does not cover validation and final release of the medical device, even when the medical device consists entirely of software. As provided in the standard, validation is a system level activity, and not a software specific activity. Therefore, software testing alone is not enough to say that software is validated.

According to Clause 14 of the **IEC 60601-1** standard, a PEMS Validation Plan shall include the validation of basic safety and essential performance, and shall require checks for unintended functioning. The person responsible for the validation must be independent, and development team members cannot validate their own software.

**IEC 62366** and **IEC 60601-1-6** require usability validation of the medical device to confirm if the acceptance criteria defined in the Usability Validation Plan have been met. The usability validation efforts depend on the criticality and functionality of the medical device. Individuals that were directly responsible for the user interface design should not be involved in the usability validation. *Usability tests* can be conducted in a laboratory setting, a simulated use environment or the actual use environment. The tasks performed by representative intended users during a validation test shall include the primary operating functions (i.e. critical functions on which a risk can be identified), use scenarios representing the frequently used functions (i.e. functions that involve user interaction with the medical device) identified in the Usability Specification, and worst case use scenarios. *Risk analysis* technique should be used in addition to usability testing to detect low probability use errors. Validation of medical device software should address human factors in system operation (e.g. GUI, warning and error messages, alarm signals, reports, help files, user manual, etc.) and collect feedback on potential usability issues from representative intended users.

### *Traceability*

Full traceability throughout the software development life cycle shall provide an auditable trail of documentation that links requirements to specifications through to the documented results of the verification and validation activities. Traceability gives confidence that the affected product components can be identified, if requirements change during or after development.

In the most common form of the Traceability Matrix, the input requirements are listed in a table, and references are provided to the output documents which address or satisfy each input requirement (e.g. (software) system requirement -> software requirement -> software design -> test plan -> test case -> test report). Traceability between risk control measures (i.e. implemented in hardware, software, and/or labeling), input requirements, and verification and validation activities, shall also be considered.

### *Conclusion*

Sophisticated and regulated life science industries (i.e. medical devices, healthcare services, biotechnology and pharmaceuticals) need safe and reliable products with high quality. Quality of medical device software cannot be achieved by software testing alone. Well planned, performed and documented V&V activities strive to ensure that quality is built into the software product from the start, and that the software product meets all requirements and user needs. Preventing and detecting errors in the early phases of the software design and development process also reduces development costs and time.

Adequate usability ensures that the risks resulting from normal use are acceptable. Therefore, increased and early involvement of users in the product design and development is very important to detect use errors. Intuitive and user-friendly product design requires less support and training time, contributes to the competitive advantage and results in a product that will probably survive market competition longer.

Failure risks of medical device software are low if good software design, development and maintenance processes/best engineering practices as well as applicable standards and regulations are followed.

### *References*

[1] ANSI/AAMI SW68:2001 Medical device software – Software life cycle processes

[2] Code of Federal Regulations Title 21 Part 801 Labeling

[3] Code of Federal Regulations Title 21 Part 803 Medical Device Reporting

[4] Code of Federal Regulations Title 21 Part 807 Establishment Registration and Device Listing for Manufacturers and Initial Importers of Devices

[5] Code of Federal Regulations Title 21 Part 812 Investigational Device Exemptions

[6] Code of Federal Regulations Title 21 Part 814 Premarket Approval of Medical Devices

[7] Council Directive 93/42/EEC of 14 June 1993 concerning medical devices

[8] Council Directive 90/385/EEC of 20 June 1990 relating to active implantable medical devices

[9] Directive 2007/47/EC of 5 September 2007 amending Council Directive 90/385/EEC, 93/42/EEC and Directive 98/8/EC

[10] IEC 60601-1-4:1996 + A1:1999 Medical electrical equipment – Part 1-4: General requirements for safety – Collateral standard: Programmable electrical medical systems

[11] IEEE 610.12-1990 Standard Glossary of Software Engineering Terminology

[12] ISO/IEC 12207:1995 Information technology – Software life cycle processes


## > biography


**Nadica Hrgarek**


*is a Software Quality Engineer at MED-EL Elektromedizinische Geräte, an innovative hearing implant company based in Innsbruck, Austria.*
*Hrgarek received a B.Sc. and a M.Sc. in information sciences from the University of Zagreb, Faculty of Organization and Informatics, Croatia. Her current research interests include quality management, process improvement, software process modeling, software metrics, and agile software development.*
*She is a member of the German association for software quality and training (ASQF).*




# How do finance applications/ systems test?

*by Ashish Bharanuke*

Do we have to take care of the domains?

The answer to the above questions is YES! It is always better to have domain knowledge because it allows you to look in a superior and lucid direction. It's like asking, do I need to know all about the ingredients in order to be a good cook? The answer is: of course you need to know them. Otherwise, how else will you develop the best flavor, if you don't know the effect of the ingredients you are using?

Most customers/companies always look to optimize processes for better results, so they will give high precedence to the experience and expertise in their new business enterprise, because they can add value to software testing. So one of the key factors for any success is adding Domain expertise to the project/product.

Domain expertise is important in software testing because the person who has domain knowledge can test the application better than others because he/she knows the ins and outs. The domain knowledge plays an important role in software testing as one of the software testing principle says "**Testing is context driven**", which means the testing performed for a financial application is different to the testing done for an insurance or health care application.

When we test a particular product, we have two kinds of axis where the Y axis denotes the technology and where the X axis denotes the domain. The domain is common for all technologies. Domain experts can write the business driven scenarios to execute the actual business, whereas technology experts will be able to write scenarios on the technology which makes their scope narrow.

In this article, I have analyzed how domain knowledge is important in the testing of fnancial systems or applications on the basis of the SWOT (Strength, Weakness, Opportunity, and Threats) analysis theory which will be elaborated further.

***Introduction:***

This article provides an overview of how domain knowledge is essential in the testing of any finance applications/systems.

Domain knowledge is knowledge about a specific field of interest/subject. As testing is a part of software engineering, domain knowledge knows about the environment in which target system operates.

As all of you know, finance is a science of fund management. The general areas of finance are business finance, personal finance, and public finance. Finance includes saving money and often includes lending money. The field of finance deals with the concepts of time, money and risk and how they are interrelated. It also deals with how money is spent and financially planned. Finance works most basically through individuals and business organizations depositing money in a bank. The bank then lends the money out to other individuals or corporations for consumption or investment, and charges interest on the loans.

Types of financial services include:

- Bank and banking Services (includes other type of services such as private banking, capital market bank, bank cards, credit card machine services and networks)
- Foreign exchange services (includes currency exchange, wire transfer, foreign currency banking)
- Investment services (asset management and hedge fund management)
- Insurance (includes insurance brokerage, insurance underwriting, and reinsurance)
- Other financial services (includes advisory services, private equity, venture capital, angel investment, and multinationals)
- Custody services

On the basis of the SWOT analysis theory we will get a more vivid picture to find an answer to the question "Do we need to have domain knowledge for testing finance systems or applications?".

SWOT analysis is a strategic planning method used to weigh up the strengths, weaknesses, opportunities, and threats involved in a project or in a business venture.

As I mentioned in the introduction, having domain knowledge is essential for testing any domain, whether it is in finance, auto-


Fig1.1. Financial services and types

motive, medical etc. . The finance domain is like a vast sea with lots of concepts, terminologies and calculations. It's all about money, so based on the SWOT theory we will analyze how this is going to be essential for software testing. Based on the diagram of SWOT analysis below (fig. 1.2), I have elaborated each point with a corresponding explanation.

## Strengths:

The strengths which give a competitive advantage over colleagues and team members are:

- **Less training time required**: A person with domain/industry knowledge can be productive quicker than a person without. This adds value to project/product.
- **Good idea of functional flow (workflow), the business processes and rules**: This will help to better understand the product requirements. What does the customer/client need?
- **Vocabulary/glossary**: Reporting in the language of the business results in good understanding with the business team.
- **Domain knowledge is also important to defect triage**: Knowing how the application will likely be used and how it is expected to perform, will help the tester in deciding whether a given defect is trivial, significant, or critical.
- **Good idea of user interface features**: This will help the person to not only improve the look & feel, but also to find more bugs in the initial testing of the user interface.
- **Good idea of back-end processing**: This is about knowing how effectively/ efficiently the data/code is handled.

Like other professionals, software testers continuously attempt to add value to their work. Provided they are able to take time with their tests, they attempt the following in order to stand out in their projects:

- Define and tune the testing methodologies in use (process improvement)

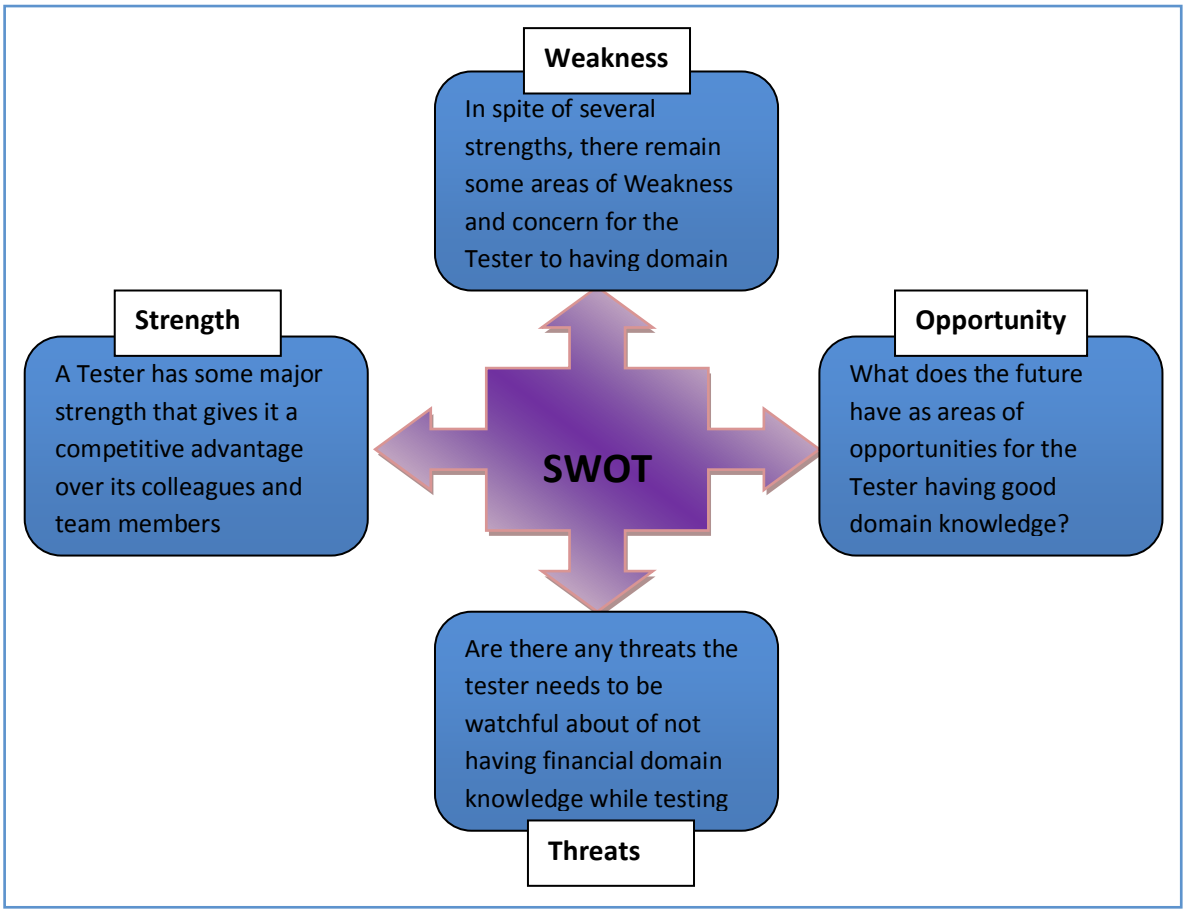



Fig 1.2 SWOT analysis

- Find out and use any automation tools to provide more/regular test coverage or speed up their existing tests (new or more test automation)
- Report test results to the stakeholders in a more useful way (better communication)

One improvement area that is overlooked by many testers is their domain (business) knowledge related to their project. In this article, I would like to draw your attention to this point.

Professionals from many backgrounds work in the software testing field. Some professionals have vast experience in the testing field. They are aware of many ways to test, they might have good people skills and reporting skills and they might be aware of many resources and tools that they could leverage in their tests. Some other professionals are ex-developers and they bring their technical acumen to testing. They usually have rich experience in at least one development environment, they are quick to conceptualize the implementation of the software they test and they can learn new test automation tools very quickly. There is a third category of professionals who work in the software testing field. They come from the business background. They have been in the industry for a long time and they are keenly aware of the general requirements of the business and the end-users. Of course, these professionals from the business background might not know a great deal about testing methodologies or test automation tools, but they know the business domain.

If you work in the software testing field and you do not belong to the third category of professionals, you should know that building up your domain knowledge would put you ahead of many colleagues and fellow testers who do not choose to build that knowledge actively.

It is extremely important that software testing teams have a good understanding of the business processes which are implemented in the application or system. For this the testing team needs to be trained/oriented towards the business flow so that they can design the test cases which are significant from the business perspective. The objective should be that testing becomes the customer's voice in the organization. It's about understanding customer usage patterns.

The customers often use the product in a different way to the way we use them during development and testing. This results in customers encountering important defects which went undetected in the test cycle. It is therefore important to understand how the customer actually uses the product. If possible, there should be an opportunity for the test team to interact with customers and watch them using the product.

## Opportunities:

Having good domain knowledge is very important in any field of information technology, be it business analyst, development, software testing or support, because information technology is extensive in a variety of domains & fields. Also, the application of information technology is different in different domains, e.g. the concepts in the financial domain will not be present in the medical, automotive or telecommunication fields. So as a software tester, it becomes even more important to understand the financial application domain and the trends of the industry you are working for. The software engineer has a direct interaction with many different groups such as development teams, business analysts and also with customers/clients. So to understand the requirements and create optimized test scenarios, it's important to also understand the relevant domain. It also helps in understanding the technical terminology of that industry, which helps in better communication with business analysts and customers. Domain knowledge and expertise not only save training and ramp-up time, but also helps the software test engineer's team to come up with a more effective test plan, which may also include the user interface, security, load and end-to-end test scenarios/test cases in accordance with the needs of the financial domain. Eventually, it helps in better understanding of the whole system, leading to the best possible testing and high quality end products which justify the purpose of testing.

So based on the benefits described, a software test engineer with good financial domain knowledge can pursue his/her career into the following areas:

- Business analyst
- Subject matter expert
- Architecture group
- Consulting services for banks and client organization (financial and non-financial institutions).

***Weakness:***

- How you will come to a conclusion about defects?
- The tester will be definitely able to test the product according to the requirements, but he/she won't be able to test it independent of the specifications (like error guessing and the calculation for transactions & interest rates). A tester with limited or no domain knowledge will be the opposite of the above and will therefore be less effective than a tester with domain knowledge.
- One way to consider the matter is the difference between someone receiving directions in their home town versus receiving directions in a town they have never before visited. A person receiving directions in their home town would be much more likely to catch any minor mistake, such as mistaking "Park Street" for "Park Road."
- Likewise, the programmer with domain knowledge will be more likely to understand the gist of the requirements. For example, if a requirement is accidentally specified with a wrong variable in an equation, the programmer with domain knowledge will be likely to catch the error before starting the programming. Fixing errors at that stage is cheap and easy. If the programmer does not have domain knowledge, such an error might not get caught until testing or even deployment. The error thus becomes more costly.

## Threats:

The project schedule will get hampered. For example, if a graduate is asked to develop/test a finance project, then he/she might take the first few weeks just to understand the terms used in the requirements document.

***Conclusion:***

When summarizing the analysis, which has more strengths and opportunities than weaknesses and threats, it becomes obvious how important domain knowledge is.

Let us now come to the various ways in which you can increase your domain knowledge pertaining to your project/product. Many software test engineers who have not worked in prior pro-

jects related to an industry, e.g. investment banking, retail and so on, learn about the project from the available documentation and the knowledge given to them by their project team members. The documentation may exist in the form of a requirements specifications, design specifications or user stories. The project team members may pass on knowledge in meetings or informal interactions. However, the initial learning from the team and continuous learning from the feedback given by the business are not the only ways to get domain knowledge.

You should also consider other ways of building up your domain knowledge:

- Show interest in your domain.
- Put your hands on whatever you have learned in past projects. If possible, work only in projects related to a particular industry.
- Go through the finance publications (articles, news items, web casts and so on).
- Increase your network with friend/colleagues who are in the same domain.

## > biography

**Ashish Bharanuke** *has been working in the information technology field for 3.5 years. He currently works as Associate Consultant at Capgemini India Pvt Ltd located in Pune, having a variety of knowledge of manual and mainframe testing in the banking and finance domains. He is a Science graduate from Mumbai University pursuing his MBA finance as a correspondence course.*

# 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

350.000 readers

ADVERTISEMENTS
sales@testingexperience.com

SUBSCRIBE
www.testingexperience.com/subscribe.php

PRICE
online version: free of charge -> www.testingexperience.com
print version: 8,00 € (plus shipping) -> www.testingexperience-shop.com

ISSN 1866-5705



# Index of Advertisers

Berlin, Germany

# IT Law
# Contract Law

German
English
Spanish
French

www.kanzlei-hilterscheid.de
info@kanzlei-hilterscheid.de

k a n z l e i  h i l t e r s c h e i d

# Training with a View

Díaz Hilterscheid

| | | |
|---|---|---|
| 14.03.11–17.03.11 | Certified Tester Foundation Level | Berlin |
| 21.03.11–25.03.11 | Certified Tester Advanced Level - TEST ANALYST | Berlin |
| 29.03.11–31.03.11 | Certified Tester Foundation Level - Kompaktkurs | Frankfurt |
| 30.03.11–30.03.11 | Anforderungsmanagement | Berlin |
| 04.04.11–08.04.11 | Certified Tester Advanced Level - TESTMANAGER | Frankfurt |
| 05.04.11–07.04.11 | ISEB Intermediate Certificate in Software Testing | Berlin |
| 06.04.11–08.04.11 | Certified Professional for Requirements Engineering - Foundation Level | Mödling/Austria |
| 11.04.11–14.04.11 | Certified Tester Foundation Level | München |
| 11.04.11–12.04.11 | Testmetriken im Testmanagement | Berlin |
| 11.04.11–15.04.11 | Certified Tester Advanced Level - TECHNICAL TEST ANALYST | Stuttgart |
| 18.04.11–19.04.11 | Testen für Entwickler | Berlin |
| 03.05.11–05.05.11 | Certified Tester Foundation Level - Kompaktkurs | Mödling/Austria |
| 09.05.11–13.05.11 | Certified Tester Advanced Level - TESTMANAGER | Mödling/Austria |
| 12.05.11–13.05.11 | HP Quality Center | Berlin |
| 16.05.11–19.05.11 | Certified Tester Foundation Level | Berlin |
| 16.05.11–20.05.11 | Certified Agile Tester - CAT \| english | Berlin |
| 23.05.11–27.05.11 | Certified Tester Advanced Level - TESTMANAGER | Berlin |
| 30.05.11–01.06.11 | Certified Tester Foundation Level - Kompaktkurs | Mödling/Austria |
| 06.06.11–10.06.11 | Certified Agile Tester - CAT \| english | Berlin |
| 08.06.11–09.06.11 | HP QuickTest Professional | Berlin |
| 14.06.11–16.06.11 | Certified Professional for Requirements Engineering - Foundation Level | Berlin |
| 14.06.11–16.06.11 | Certified Tester Foundation Level - Kompaktkurs | Mödling/Austria |
| 20.06.11–24.06.11 | Certified Tester Advanced Level - TESTMANAGER | München |
| 20.06.11–22.06.11 | Certified Tester Foundation Level - Kompaktkurs | Berlin |
| 27.06.11–01.07.11 | Certified Tester Advanced Level - TEST ANALYST | Berlin |
| 04.07.11–08.07.11 | Certified Tester Advanced Level - TESTMANAGER | Berlin |

- subject to modifications -

more dates and onsite training worldwide in German, English, Spanish, French at http://training.diazhilterscheid.com/

iseb Accredited Training Organisation

IREB International Requirements Engineering Board

CAT Certified Agile Tester®

ISTQB® Accredited Training Provider

Kurfürstendamm, Berlin © Kai-Uwe Schülke