5

March, 2009

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Outsourcing

Agile TESTING DAYS

Berlin, Germany

Call for Papers

October 12
Tutorials

October 13
Conference

October 14
Conference

The Agile Testing Days is the European conference for the worldwide professionals involved in the agile world. We are very proud to count with the support and the work of Lisa Crispin, Elisabeth Hendrickson, Tom Gilb, Stuart Reid, Isabel Evans, Alessandro Collino, Mary and Tom Poppendieck.

We decided to choose Berlin as one of the best connected capital in Europe and also for its very good price/service ratio for hotels and flights. Berlin offers also a lot of fabulous places to visit in your spare time.

Please follow the call for papers and send us a proposal based on your experience. Come and enjoy the conference!

Call for Papers - send your proposal by March 30!

www.agiletestingdays.com

Our Programme Committee


Lisa Crispin


Elisabeth Hendrickson


Alessandro Collino


Tom Gilb


Stuart Reid

Our Tutorials on October 12th


Lisa Crispin


Elisabeth Hendrickson



Stuart Reid
Isabel Evans


Alessandro Collino


Tom Gilb



Mary Poppendieck
Tom Poppendieck

Dear readers,

If you want to have an amazing evening, invite people to talk about outsourcing. There are good and less good stories about it, with a lot of pros and cons. In my opinion, and I think that you can follow it in the articles, the outsourcing of services is most of the time possible and it can bring you the expected ROI, if you pay attention to risk requirements, aims, culture, costs, time etc. And all of it embedded in a good project management frame. Experience does a lot! If you don't do so, then you may get the opposite.

When I was in India last October I had the chance to look at PureTesting's Testing Center. I was amazed about the quality and the way they work. I think that one of the important things in making a decision for outsourcing is to have the appropriate partner. In India and other countries there are a lot of well skilled and culturally aware companies, which could help you if you decide to go this way.

On the other hand there are a lot of fears about losing jobs – especially in this crisis time – due to the fact that companies see outsourcing as a part of the business where they can save money apparently quickly. In this case I think that there is only one thing you can do: be real added value for your company. If they decide to go to a another country to let people do your job, then you have to improve your output and not be considered a cost factor but an added value. I get mad when I speak with customers or test managers and they talk about thousands and thousands or hundred thousands or millions of test cases. No every customer developes an operating system, flight control system or similar. Who should test that? Are you really a professional tester? How could this happen? Do you know test techniques? Well, with that amount of test cases it is quite normal that they decide to move your position to e.g. China!

We are very proud of having over 200,000 downloads with the fourth issue. It is amazing. Thanks. We have sent the printed magazines to all registered people around the world. We could do that because we had enough ads to earn the money to do it. The number of registered persons increased a lot with the last issue and we cannot now afford to send the printed issue out to all using the money earned from the ads. We have therefore decided to send it only to the those who pay for it. The annual fee is 32 Euros. We think that's a fair price. Please go to the website www.testingexperience-shop.com to register for the printed issue. You can pay with PayPal or we can send you an invoice. The online issue will be always available for free!

We did have a very successful **Belgium Testing Day**. We were around 100 professionals there enjoying the talks. We will have the next one on February 8th, 2010. Save the date!

Last but not least something about my holidays: In Gran Canaria I met Celeste from Italy and George from UK. We went for a dinner at the beach promenade and we did a little sightseeing in Las Palmas. I enjoyed the evening. On January 2nd my son and I jumped in to the water. The Atlantic! It was wonderful. We had 26°C!!! When we came back to Germany we had -16°C. What should I say…

I will be in Sydney at the ANZTB Test 2009 conference. Let's go for a coffee!!

Enjoy the magazine and thank you very much for your support.

Yours sincerely

José Manuel Díaz Delgado

Contents

Editorial 3

How to Build and Maintain a Successful Outsourced, Offshored Testing Partnership 6
by Dr Mike Bartley

Outsourcing; what does it mean for us? 11
by Erik van Veenendaal

The NEW Economics of Global Outsourcing 12
by Jerry E. Durant

Outsourcing And Unit Testing 16
by Jean Joskowicz

A desire for cost cutting is not the best starting point for offshoring 19
by Rik Marselis & Holmer Vonk

Mid-Term Health Check 22
by Nishant Pandey

Us and them: when cultures collide 25
by Jos Paeshuyse

Security Testing and Risk Management 28
by Eric Jarvi

»Let's Talk About Testing« 34
by Andreas Spillner

Interview Doron Reuveni 38

Requirements-Based Testing - Cause-Effect Graphing 40
by Gary E. Mogyorodi

Thinking Outside Of The Box 45
by Aleksandra Popara & Predrag Skokovic

To be or not to be 48
by Todd E. Sheppard

Intelligent Use of Testing Service Providers 50
by Rex Black

What happens to usability when development goes offshore? 57
by James Christie

The impact of the global financial crisis on offshore outsourcing 64
by Nadica Hrgarek

Index Of Advertisers 67

Interview Dr Mike Bartley 68
MBT as the next step in testing! 69
by Elise Greveraars
Interview Sumithra Gomatam 73
Co-Shoring – All The Benefits Of Onshoring With The Competitive Costs Of Offshoring 75
by Osmar Higashi
Quality, Excellence and Cost Effectiveness 78
by Chen Bressler
Command Line Testing With The Robot Framework 80
by Alessandro Collino
Agile Specification Quality Control: 87
by Tom Gilb
The Reality of Software Testing in an Agile Environment. 94
by George Wilson
Beyond Functional Testing: On to Conformance and Interoperability 98
by Derk-Jan de Grood & David Bakker
Masthead 102
Outsource your testing – Is it really worth it? Some tips... 103
by Yaron Tsubery

How to Build and Maintain a Successful Outsourced, Offshored Testing Partnership

by Dr Mike Bartley

Somebody recently asked me to review a course on "Building a Great Team" - where is the section on outsourcing? I asked. In my opinion, outsourcing should always be a consideration when trying to maximise the performance of a team. In this article I will consider this in the context of software *testing*. I will mainly consider software testing that is both offshored and outsourced[1], although it should also be applicable to software *development* and to companies that just use outsourcing or offshoring. The article is based the successful application of outsourcing at ClearSpeed Technologies plc (which I've described elsewhere -see [1]) and in advising other companies on how to outsource their software development and testing.

The objectives for offshoring are evolving. Initially companies looked for cost savings. Then companies looked externally for extra resource (for example during the Y2K crisis). Companies are now also looking for strategic benefits and better defined business impacts from their offshoring relationships. My aim in this article is to demonstrate how such benefits and impacts can be achieved with minimum risk.

To attain the strategic benefits you seek, you must first identify them. Let's consider the benefits claimed for offshoring by managers of technology SMEs (Small/Medium Enterprises) in [2]:

To accelerate the maturation of new products: "*The internal skills of rapidly developing products that meet market needs is augmented by offshored teams who have the patience, precision and perseverance to evolve young products into mature ones*" (Jajiv Dholakia, Cenzic Inc.)

Agility and cost savings: "**on average we are able to deliver projects at a much faster rate and save at least 30% cost to our clients**" (Jing Liu, CEO, Enter Suite)

Better software through additional resources: "*The additional talent and resource that we can carry allows us to develop better software than our competition can afford in the same period*" (Scott Allan, Symbol technologies Inc.)

Concentrate on core competence: "*Businesses should focus on clients and IP which satisfies their need. Everything else should be outsourced to those whose core competency is that business*" (Bill Widmer, President and CEO, g8solutions)

Ability to undertake one-off projects: "**we were entering a virtually untested market with a new product. The US team focused on the current roadmap while I was able to develop the new product at a rapid speed and at a low cost**" (Gopan Madathil, Founder, TechCoire)

The strategic goals must also be agreed by a suitable forum. When I started on the offshoring route at ClearSpeed, I (as the Quality Manager) agreed the following prioritised objectives with the CFO and Engineering Manager.

- To improve company focus.
- To improve company resource flexibility and by doing so, reduce time to market.
- Quality improvement.
- Cost variablization[2].
- Cost reduction.

The above goals were shared with the offshore supplier. This gave us a common understanding which helped the day-to-day management of the projects. We also derived metrics to measure our progress against the goals. For example, we used Defect Detection Percentage (DDP) to measure the improvement in our testing (see [3] for an explanation of DDP). The time to market and the cost of testing relative to development were also tracked.

The next major decision regards on your road to offshoring is to decide what type or stage of testing to offshore. There are a number of considerations to take into account.

- **Amount of documentation:** In order to test your software your offshore supplier will need to understand it. This will ultimately require good documentation.
- **Domain knowledge:** How much domain knowledge will the tester need and how easy will it be to find or train testers with that knowledge?
- **Complexity of the environment:** How complex is the test environment and how easy will it be for the offshore test organisation to re-create it?
- **Development process:** Testing within the V model lends itself well to offshoring. Offshoring of agile development less so as it is based on nimble teams with good communication and rapid change.
- **Stability:** How stable is the software that you are asking the supplier to test.
- **Level of interaction:** To what extent will the offshored test team need to interact with the development team.
- **Level of independence:** NASA uses an independent V&V team to improve the

1 Note offshoring does not imply outsourcing – a company can establish an overseas function.

2 This means converting fixed costs to variable costs.

quality of their software (see [7]). An offshore test team allows you independence along three dimensions: managerial, financial and technical.

- **Taking a customer perspective:** An offshore team can be used as a "friendly" customer, helping to identify all the problems that your real customers would otherwise find.
- **IP protection:** This is always a major concern in outsourcing and the concern seems to deepen with offshoring.
- **Sign-off criteria and incoming acceptance:** You will need to sign-off the work performed by your supplier through some sort of incoming Quality Assurance (yes – testing the tester!).

Now consider how the above affect your decision on what to outsource:

- Unit testing:
  - This will probably require you to release source code which may lead to IP protection issues.
  - Unit code may not initially be stable and may not be well documented.
  - The offshore organisation can act as an independent V&V team.
  - The test environment for unit testing is not usually over-complex and the domain knowledge requirements tend to be lower.
  - Sign-off could be defined through structural coverage goals.

As an example, I successfully offshored the unit testing of a new library of memory management functions. The functions were developed using agile techniques in close collaboration with marketing and two lead customers and, once the specification was stable, we offshored the unit testing. We had an existing, trusted relationship with an offshore partner who first produced a test specification and then tests against that specification. Finally, they topped up their tests to ensure our structural coverage goals were met. Our incoming signoff consisted of signing off the test specification, reviewing a selection of actual tests against that specification and re-running the tests to check the coverage.

- System testing:
  - In my experience by the time you reach system testing the software is more stable and the chances of some documentation are improved.
  - If you are following the V-model of development then the appropriate documentation may have been available for a while so that the offshore organisation could have already written test specifications reviewed by you.
  - The test environment and domain knowledge required can be complex.
  - The level of interaction should be reduced compared to earlier stages of testing.
  - The offshore team can perform the testing independently and can take a customer perspective in their testing.
  - IP protection issues can be avoided by releasing the binaries rather than source.
  - Testing the tester is harder as an objective measure of system test quality is more difficult - but it is not impossible. A system test specification and reviews of the test scripts for example. The test results should be also made available for audit.

  Some of my most successful offshored testing has been in system testing at ClearSpeed. We had an automated test environment which gave very high unit and integration test coverage, and some system testing. We then offshored all of the system testing that was impossible or not cost-effective to automate. The offshore team was encouraged to act as a customer with no prior assumptions and not to use any work-a-rounds offered by "helpful" members of the development team!

There are numerous other types and phases of testing where the above considerations can be applied but space does not allow for here. For example: **specialist testing** (such as security testing) where the outsourcing arguments are very strong but expertise rather than price will be most important; **compliance testing** (where you are required to demonstrate conformance to a particular standard) where there is a strong argument for outsourcing to specialists in that standard; **release or acceptance testing** has similar arguments to system testing discussed above. And the list goes on…

You also need to identify your preferred outsourcing model. [5] describes three types of service deal related to the scope and complexity of the offshoring contract and your strategic objectives.

Efficiency: These deals tend to focus on cost efficiency, improved resource agility or greater organisational focus on core skills. The contract terms concentrate on service levels. So, for example, you may be outsourcing unit testing currently performed by developers to free them up for additional projects and the service level

Commercial model	Description and discussion of risk	Recipient Risk	Provider Risk
Cost-plus pricing	The recipient pays the providers costs plus a percentage. The provider is guaranteed cost recovery. The recipient is not guaranteed service levels. Appropriate for efficiency deals.	Medium	Low
Fee-for-service pricing	Price is based on the amount and/or quality of the work actually delivered. The provider's costs are not covered unless service levels are met. The recipient costs are not entirely predictable and service levels are not guaranteed. Appropriate for efficiency and enhancement deals.	Medium	Medium
Fixed price	Defines a specific service level and a price for achieving it. This is higher risk for the provider. The recipient has lower risk as prices are capped. Appropriate for efficiency deals.	Low[3]	High
Shared-risk/reward pricing	This involves a flat rate with additional payments for achieving specified outcomes. Appropriate for enhancement and transformation deals.	High	High
Business outcome achievement	The provider only receives payment for achievement of specified business outcomes. The receiver has no guarantee that expected outcomes will be achieved. Appropriate for transformation deals.	Medium	High

Table 1: Commercial models, service deals and risk

3 I have been involved in too many fixed-price deals to agree with [5] that this is low risk for the recipient. There is always the risk of changes in requirements which can lead to significant cost increases.

might be around test specification, structural coverage and time to completion.

Enhancement: These deals are more to do with process improvement. For example, you may outsource to a specialist test company to improve your security testing.

Transformation: These deals aim to improve your competitiveness through innovation in the relationship. So, for example your goal may be to reduce time to market or improve the quality of your software (through reduced DDP for example). Your outsource supplier would need to act in a consultancy role as well as a service supplier.

Obviously the above three imply an increasingly complex relationship with increasing levels of trust. [5] goes on to consider the appropriate commercial models for the deal and this is summarised in Table 1.

You also need to consider your outsource location: on-shore, near-shore or offshore. This decision should be made in the context of your chosen service deal. I would also recommend considering multiple suppliers. In my experience this brings a number of advantages.

- Using suppliers with different types of skills.
- Using one supplier for an efficiency deal and another for an enhancement or transformation deal.
- Allowing the suppliers to competitively tender for new work (rather than becoming increasingly dependent on a single supplier).

Now that you understand what you want to offshore and your preferred commercial model, the next consideration is your readiness for offshoring. A number of attempts at offshoring fail because an organisation has bad existing internal development practices: *offshoring your chaos will just generate additional chaos!* So, how do you measure your readiness? There are formal measurements such as CMMi (see [4]). Table 2 is taken from [8] and shows the national differences in CMMi take-up. It is also interesting to note from the table the differences in maturity from the countries more likely to be supplier to those more likely to be provider. In my experience providers tend to go for CMMi for a mixture of technical, management and marketing purposes.

Most organisations will not want to gate their offshoring on such formal appraisals. In my experience, the minimum set of processes that your organisation will require to outsource software testing is:

- Requirements management
- Change tracking and management
- Project planning, monitoring and control
- Configuration management
- Build, test and release automation
- Validation and verification processes

You now need to select your partner. There is no magic in the selection process:

1. Identify potential suppliers.
2. Define your selection criteria.
3. Create a short-list.
4. Perform an in-depth appraisal of your short-list (including a visit if possible)
5. Contract negotiation.
6. Perform a pilot project with your selected supplier (or suppliers)

The major issues in the above list are identifying appropriate selection criteria and exchanging contracts. Below are some example selection criteria

- **People:** Skills profile; Academic background of staff; Staff training statistics.
- **Retention:** Staff attrition rates.
- **Process:** Process maturity; CMMi and other process maturity measures attained. You should consider how well matched the relative process maturity between your two organisations are. For example, do the outsource organisation processes require deliverables and maturity on your side that you cannot deliver on?
- **Quality:** Quality processes employed; how well do they match your quality processes?
- **Project and risk management:** What management processes do they follow? How well do they fit with your preferred management methodology?
- **Company profile:** Financial stability; historic financial data strategic goals for the company.
- **Technology:** Domain knowledge/experience of the technologies employed by your company.
- **Cost:** What are the typical pricing models? How are changes managed from a cost point-of-view? Recent cost change history.

If you are offshoring then your selection criteria will need to reflect circumstances in the providers' country. For example, if the destination is India then staff turnover or retention rate should be a major consideration.

Regarding contractual negotiations, I have found it helpful to have two documents: a Master Services Agreement (MSA) and a Service Level Agreement (SLA). The MSA covers the legal, financial and contractual issues (such as confidentiality, ownership of deliverables, assignment of rights, indemnification, warranties, liability, changes in personnel, subcontracting, termination, penalties for not meeting agreed service and delivery targets, etc.). The Service Level Agreement (SLA) should cover the service and support goals and is strongly affected by the type of service deal that you have (efficiency, enhancement or transformation). The MSA and SLA are complex legal documents which I will not attempt to cover in detail in this article.

You are finally ready to execute your offshored software testing project and this is where it starts to becomes difficult! If you have followed the process I have described above then your chances of your success are significantly improved. However, any offshored development is inherently risky. First there are the problems of managing a distributed team (communication issues, knowledge management, change management, etc). Then, according to Ralph Kliem in [7], in offshoring there are also "*the unique challenges posed by geographical, cultural, and other differences*". There is not enough space in this article to go into details of how to manage offshored projects but I will highlight the main items to consider. Firstly, you need to distinguish management and governance. Management is about responsibility for specific decisions; Governance covers decision-making processes: the who, the how and the when. It is important to define the governance rules to allow your offshore team to know when they are empowered to make decisions and how to make good decisions.

Country	Total Appraisals	Maturity Level Reported				
		1	2	3	4	5
Argentina	47		31	10	2	3
Australia	29	1	8	4	2	4
Brazil	79		37	31	1	8
Canada	43		10	18	5	3
China	465	1	103	293	18	34
Egypt	27		12	11	2	2
France	112	4	67	34	1	2
Germany	51	7	27	1	1	1
India	323	1	11	127	22	151
Japan	220	16	64	88	13	15
Korea, Republic of	107	1	31	48	11	7
Malaysia	42		15	24		3
Mexico	39	1	18	13	3	4
Spain	75	1	49	21	1	3
Taiwan	88		60	25		2
UK	71	3	36	24	1	2
US	1034	25	365	347	21	114

Table 2: Number of Appraisals and Maturity Levels Reported to the SEI by Country (where total appraisals is above 20)

Good governance involves at least the following

- Established standard decision-making procedures.
- Well-understood processes for handling exceptions.
- Decisions made quickly at the correct level against well-defined criteria.
- Decisions get recorded.

And good governance is helped by the following

- Having agreed strategic objectives.
- Formal communication methods which record project status and decisions.
- Controls & records to ensure governance procedures are followed.
- Regular meeting of the stakeholder group.

Good management means the usual good software project management practices plus

- Communication
  - Who can communicate with whom? And how often?
  - Formal vs. informal communication.
  - How does the communication get recorded?
  - What happens on a miscommunication?
- Knowledge management
  - Getting up the initial learning curve and staying there through staffing changes
  - On-going knowledge management
    - What is the process for asking a question?
    - What is the process for recording an answer?
- Audit trails
  - What trails do you need? Decision making, Execution history, Timesheets?
  - You need to save this data quickly and cheaply but what about access time?
  - How strict does your audit need to be? Are their any legal requirements?
- Well-defined incoming QA procedures

In summary, there are many benefits to offshoring your software testing, especially if you aim to build a competence based rather than transactional outsourcing relationship. However, the risks are also high and in the above I have laid out a process to improve your chances of success.

- Agree the strategic goals in an appropriate forum and share them with your provider.
- Decide what type or stage of testing to offshore.
- Identify your preferred outsourcing model and location.
- Assess your readiness for outsourcing.
- Select your partner or partners.

References

1. "How to Boost your Productivity through Outsourcing", Mike Bartley, Software and Systems Quality Conference, London, Sept 2008
2. "Happy About™ Outsourcing", Mitchell Levy, January 2005
3. "How to measure test effectiveness using DDP (Defect Detection Percentage)", by Dorothy Graham et al, Grove Consultants (available from http://www.dorothygraham.co.uk/downloads/generalPdfs/DDP_Tutorial.pdf)
4. Capability Maturity Model Integration, http://www.sei.cmu.edu/cmmi
5. "Multisourcing: Moving Beyond Outsourcing to Achieve Growth and Agility", Linda Cohen and Allie Young, Gartner, Inc.

Biography

The author, Dr Mike Bartley, gained a PhD in Mathematics at Bristol University and an MSc in SW engineering and MBA with the Open University. More recently he has added ISEB Practitioner Testing Qualifications in both Test Management and Test Analysis. Mike has been involved in both software and hardware development for the past 20 years, including outsourcing, and regularly writes articles and delivers conference papers. He has recently established his own consultancy to help companies in product QA and offshoring.

CONQUEST '09

12th international Conference
on Quality Engineering in Software Technology

Nuremberg, Germany
16 – 18 September, 2009

YOUR CONSTRUCTIVE CONTRIBUTION IS NEEDED!

CALL FOR PAPERS

submission deadline is 31 March

We accept articles on the following topics

- Quality Assurance and Management
- Process and Product Quality
- Process Optimization and Improvement
- Education, Training and Certification
- Quality Engineering Related Standardization
- Systematic Software and System Development
- Secure and Safe Software-Based Systems
- Model Driven Engineering
- Requirements Engineering
- Verification and Validation
- Testing of Software-Based Systems
- Metrics and Measurements of System Quality and of Development Processes
- Analytical Models of Software Engineering
- Project Management

iSQI®
International Software Quality Institute

www.conquest-conference.org

Organizer . iSQI International Software Quality Institute . Am Weichselgarten 19 . 91058 . Erlangen . Germany
Tel +49 9131 91910-0 . Fax +49 9131 91910-10 . www.isqi.org . info@isqi.org

Outsourcing; what does it mean for us?

by Erik van Veenendaal

I'm still struggling with outsourcing, or should I say offshoring? What is its real business value; is there business value? There are many conflicting reports. It is certainly more than just a trend. Most large financial institutes, industrial companies and ICT companies are at least considering some kind of outsourcing. Quite often they already have some practical experiences; some are just in the transition phase and others have already been outsourcing to, for example India, for over 10 years. Although one often hears complaints, I by now also know of many success stories. Whether we like it or not: outsourcing/offshoring is inevitable!!

What does outsourcing mean for the professional tester and for the testing process? A lot, but perhaps slightly less than for the professional software engineer. The number of software engineers will go down in the US and Europe over the next decennia. However, the need for specialists in areas such as requirements engineering and architecture will grow. Of course outsourcing implies that some day (…..) the product will return, meaning that acceptance testing will certainly 'survive'. Not only survive, it will become essential and more formal since it will be a main input to discharging the party that developed the outsourced software system.

Understanding the Culture

Working with companies from India, China, Singapore, South-Africa or Eastern-Europe means above all collaboration with people with a different cultural background. We should start to study the way we best co-operate and not just impose our processes on "them" and then be surprised the outsourcing didn't work. Many companies nowadays send their employees involved in an outsourcing process to workshops and training to understand the different cultures. Perhaps the biggest challenge is in this area; more and better communication skills are needed for ICT-staffing. Not all ICT-staff possess these skills by nature.

Agreed Requirements on Development Testing

In addition to outsourcing the development of software systems, part of the testing of the software (at a minimum component testing) is also outsourced. Often this happens implicitly without strict and formal requirements for component testing. In many cases, this is already changing; requirements are being set for the test process of the party that develops the outsourced software system. For example, test logs and (code) coverage measurements shall be reported. Also test process improvement and formally reaching a certain TMMi level will become more important for these companies. Already there is a huge interest in the developments regarding the TMMi Foundation from countries such as India. Outsourcing will only be done to companies that possess a certain CMMI or TMMi level. It is no guarantee for success, but it certainly will help. (By the way it does help, to say the least, if the company doing the outsourcing has some level of maturity as well. You cannot just outsource your problems away …….). We have to learn how to define meaningful and strict requirements for development testing, to improve the co-operation between parties. Having a universal test terminology from ISTQB, the ISTQB Glossary, will support this.

Step by step Outsourcing

Although many companies also "discuss" outsourcing integration and system testing (test outsourcing), I'm not always convinced regarding the added value. If outsourcing software development is not always successful, isn't it at least sensible to get some more positive experiences on this before outsourcing the quality assurance process as well? (Of course, there are exceptions such as outsourcing an automated regression test.) Do things step by step is my advice, don't go too fast. Of course the outsourcing service providers will tell you a different story, but looking at my own real-life working environment I see many companies that have been working with outsourcing for years. Those that are most successful are most often still in charge of integration – and system testing. For them this is only the second step in the outsourcing process. These are also core skills that need to be developed right now. How much do most testers know and understand about integration testing?

Core Competences

To me, requirements, architecture, integration, and black-box testing are the core competences that need to be further developed and the ones in which one needs to become "world class". If we can do these things well, we can successfully outsource the actual development. I now see many companies taking outsource and/or offshore decisions without a thorough requirements process. And of course why not outsource the system test at the same time …..? Unbelievable!! And later on they start complaining about it not meeting the expectations. Let's first make sure we are mature enough to outsource and are able to control what we outsource. Sometimes this will lead to a different decision. See you at a requirements engineering course ……

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

The NEW Economics of Global Outsourcing

by Jerry E. Durant

There is a profound and impassioned stigma associated with outsourcing. For companies it offers hope and promise for survival but for those displaced the topic is met with strong resentment. During years of prosperity outsourcing was utilized to increase operating return-on-investment (ROI) and fill resource gaps that could not be accommodated domestically. Now that conditions are different outsourcing is viewed as a means for survival. Yet outsourcing (offshoring, near-shoring, on-shoring, retrosourcing…) continues to carry the social stigma surrounding lost jobs, wholesale relocation of operations, and even a sense of disloyalty to our native country. There are also those that view outsourcing as risky business. Despite current unprecedented economic conditions outsourcing is still viewed as far too risky an option to consider.

For establishing proper context the topic of outsourcing will encompass offshoring, near-shoring, on-shoring, and all of the various types of services that can be serviced outside of an organization (ITO, BPO, KPO…). Outsourcing has existed for a very long time but not always under the title as such. One could argue that even mercenaries represented an early form of outsourcing. Prior to semi-official recognition in the 1990, outsource related services were referred to in many ways; service centers, external providers, service suppliers and even contract support. Recently a LinkedIn™ discussion suggested that the term 'outsourcing' should be called something different. This probably resulted from the lingering negativity associated with the title. No matter what you call it, it's a service being provided to another with the express intent of making it economical, whether in money or in terms of delivery of value that can't be realized by one's self. The emergence of 'outsourcing' reached recognition as a result of high service demands surrounding the urgency in addressing the impact of the year 2000 (Y2K). What was a low cost side-benefit became a value added promotion for outsourcing. India was one of the first to present itself as a source of low cost services supported by a richly educated talent pool. From these early years emerged an industry, promoted by size, qualifications, and a portfolio of globally recognized companies. In the shadows hid stories of deceit, failures and cumbersome delivery of services. These situations placed at risk the potential savings that lured buyers to consider outside support.

Since that time we have seen an explosion of service providers covering virtually every country around the globe. Buyers too have looked to these sources as a way to save money, provide flexible staffing, and still provide the same level of service. While numerous risks continue to influence IT service support decisions, companies continue to recognize this form of service solution as highly attractive. Others may feel trapped by relationships that seem to provide little or no suitable way out, and as a result conclude that all is lost as it relates to economic realization. Regardless of these circumstances there is value provided by extending your business capacity through controlled commitment. Understanding the true economics of outsourcing will guide buyers (and suppliers) to a rewarding, risk overt experience.

The Basics

The simple economics of outsourcing is based on rate for service. While buying decisions are sometimes guided by region, it is based on the rate that suppliers offer that will promote specific locale interest. Based on this premise, the economics and conditions for positive outsourcing will start to increase in complexity.

Some of the leading mistakes that erode the economic benefit of outsourcing include,

- Inaccurate work effort estimates,
- Poor supplier qualification,
- Overlooking onshore engagement management requirements,
- Lack of accommodations for change,
- Excessive prescriptive direction, and
- Over reliance on contractual provisions.

Each of these errors is discussed further in later sections of this document. Note that additional study (as noted by *) will provide added insight and mastery in the various topical areas. For purposes of this article we will introduce the concept and illustrate solution examples.

Effort Estimates (*)

All potential outsource efforts should have estimates established by the buyer. It is not uncommon to find companies who rely purely on bid proposal as the test for service legitimacy. The purpose of buyer based estimation is to qualify and quantify the service initiative, serve as a basis of supplier evaluation, and determine a general reasonableness of approach. Historically, companies have great difficulty producing reasonable estimates. Inaccuracy can be attributable to a wide variety of reasons,

- Single person estimations,
- Lack of defined, lean and consistently applied processes,
- Overly complex or simplified estimation model,
- Group based confusion and controversy,
- Poor empirical estimation information,
- Real-time vs. Ideal-time confusion, and
- Date and Resource influences.

While it could be argued that these are simply factors of business, each situation is solvable and produces positive flexible opportunities for overall improvements. Pre-proposal estimates are essential in measuring the level of understanding of the service needs by the supplier. The buyer can gauge the level of understanding in the transference of information, via the Request-for-Proposal (RFP) or Request-for-Information (RFI) to the suppliers. Unexpected variations brings attention and promotes resolution before commitment decisions are made that further erode outsource value gains. This also creates a channel for an open dialog that, when exploited, will generate lasting synergy.

Processes (*)

Processes do not make system solutions, people do. But we also must recognize that the power contained in defined processes afford opportunities for skill development and delivery consistency. Suppliers with vague or non-existent processes give rise to questions about the durability and quality of the outsource service. As buyers, control must be maintained over being prescriptive vs. permissive. It is more a question of 'what' services are needed and accommodated vs. 'how' it needs to be produced/ delivered. A key to depending on this approach is placing reliance on both sound process models that the supplier may have, and the supplier themselves. Common process related problems that reduce outsourcing value include,

- Narrow applicability of defined processes,
- Non-Lean,
- Inflexible,
- Use, misuse and misguided,
- Document vs. Delivery Intense, and
- Lack of process alternatives for varying reasons.

In each case these onshore, (relating to the buyer company), issues will cause costs that reduce benefits, increase costs and create risk.

Estimation Model (*)

Everything requires balance, excesses lead to unstable and unreliable outcomes. Too much or too little formality in the estimation process increases the risk that results will be unrealistic. Estimation models that operate at a gross level tend to overlook significant areas of service need. Whereas comprehensive detail driven models create excessive unsubstantiated confidence while overlooking the overall initiative. Remember that an 'estimate' is an educated and informed prediction of an outcome that will be used to guide and control delivery. It is neither a 'prediction' of reality nor a 100% guarantee or certainty. To be discussed in a later section, 'delivery guidance (Are We Done Yet?)', estimations are used initially to measure cohesion between buyer and potential supplier, and later to guide delivery. There are several solutions that can be used to deliver more realistic and lasting effective estimates.

- (*) Group estimations utilizing such techniques as wide-band Delphi, estimation poker, Putnam's Standard Component Method (PERT), story point assignment, and Critical Chain Buffering (CCB),
- Incremental work delivery dissection,
- Task level estimation for larger service delivery components, and
- Resource balancing and delivery scaling.

In short, if you have had a history of estimation issues realize that they will not resolve themselves without purposeful attention. Appropriate consideration must be given to establishing an appropriate and reliable model for proper engagement sizing. As in the case of processes value, an estimation model will establish a climate for consistency, promote skill based mastery and serve a vehicle for improving reliable estimation results.

Estimation Information (*)

In continuation, the use of historical delivery information can serve as a useful tool for further validation of estimates. While it was initially considered useful for helping to construct estimates, many have also found it more beneficial in validating the authenticity of new estimates. This eliminates the repeating of previous estimation mistakes while adopting a more fact based approach to estimation. Problems begin to show in the areas of collection consistency and depth of detailed information. These problems give rise to doubt in new estimates which then lead to questions about historical information, ultimately raising concern over anything to do with estimates. In doing so, wild and misdirected guessing occurs that produces unrealistic expectations and creates a climate that no one will endorse.

Estimations have two purposes, the effort expectations and delivery anticipation. As buyers concern is about cost, this drives the effort required while at the same time there is a level of anticipation surrounding delivery. Delays or unexpected quality issues consume the gains that the outsourcing relationship promoted. Estimate information serve and should be viewed as a control mechanism in support of the engagement. Buyer need to recognize that the validity of the pre-project estimates, supported by historical estimate delivery will serve as a strong base for authenticating supplier legitimacy.

Real vs. Ideal Time (*)

A misconception is that the units of measure for estimates are all the same. After all a work day is the same equivalent to a productive work day, right? Does your 8 hour work day equate to 8 hours of productive result? When an estimate of effort is provided there will be someone who will take this total and attempt to predict when delivery will occur (and may even shorten that if the number isn't to their liking or suggest adding more resource to deliver quicker). The fact is that time to produce a component or a system is not free from non-productive work related aspects. When this is overlooked one is adopting an 'Ideal' time attitude (e.g. 8 hour work day) vs. understanding that there are reduce productivity (referred to as 'Real' time). While further internal study might be required, some have found this may be as much as 3 hours of unproductive time per day dedicated to such things as,

- Holidays,
- Productivity rate,
- Unexpected meetings,
- Error repair,
- Email,
- Phone Calls,
- Sick Time, and
- General disruptions.

There is also the potential for further distortion caused by unaccounted for overtime or work performed outside of the workplace. True accounting, at least for project/service control, is essential for forming an intelligent base for guiding estimation information. Otherwise, these conditions will distort delivery expectations and undermine historical estimate information.

Delivery Influences (*)

From the onset stakeholders have a healthy curiosity concerning delivery. From top management to technicians, everyone is wondering when they can expect to see product/service delivery. Whether using traditional delivery methods, characterized as the 'big bang approach', or more iterative forms utilizing incremental delivery, everyone has a healthy (sometimes unhealthy) interesting when the end will occur. Without question there will be those who will challenge whatever estimates are presented. While responsible challenging is encouraged to optimize approaches, methods, needs and increase ROI, irresponsible challenges introduce risk that will destroy the positive economics of outsourcing engagements. Misguided actions are commonly involve,

- Excessive and thinly managed resources,
- Adding more time but holding delivery dates constant, and
- Not establishing a sense of delivery importance to each.

Healthy and open discussions encourage responsible behavior and reduce risky date driven behavior. Do not take this lightly. Buyers need to adopt a responsible behavior concerning delivery in order to encourage a proper buyer attitude. Suppliers operate in a highly competitive environment and will do everything possible to gain trust and business, sometimes at the mutual peril of themselves and of their customer. For this reason a proper responsibility must be exercised. There will be ample opportunities to discuss debate and develop solution delivery that will fit the needs of buyers and service providers.

Outsourcing Gone Bad

Even with a profound understanding about the fundamental mechanics of outsourcing estimation pitfalls, there remains an element of

risk. This is contributed by issues involving transitioning of operational culture from buyer to supplier. Factors involving the mode of operation, terminology, explicit and assumed expectations and general desires require formal transference to the supplier. The supplier must then make adjustments and adaptations in order to fulfill the delivery commitment, while sustaining a stable operating environment within their company. The level of coordination and adaption is often underestimated. When failures occur it is blamed on culture and not on inattention to this important project administration point.

The first step to chipping away at these risks is to reflect on the level of project management involvement. Has proper care been given to,

- Correcting known deficiencies in artifacts, processes and operational involvement?,
- Established an orderly transitioning of duties from internal staff to external suppliers?,
- Developing sufficient checks-and-balances to retain continuity and control?,
- Adopting a 'buy right' vs. a 'buy large' attitude that insures viability and capability as cornerstones for select?,
- Clearly understanding governance responsibilities?, and
- Appropriately prepared for known contingencies (e.g. scope change)?

Contractual provisions help to arbitrary differences, and are essential, but do little to soften the effects of failed service delivery.

True Costs

As described earlier, outsourcing saves money based purely on a cost per unit (rate) of measure basis. To illustrate this point, if a company spends $50 per hour for a project manager and their benefits represent an additional 25% ($12.50/hr.) this would equate to $62.50 per hour for labor cost. This does not include office occupancy and related professional developmental requirements. This also assumes 100% utilization, which as previously discussed relative to 'Ideal Time', is impossible to attain and as a result further increases the delivery costs when serviced internally. For the purposes of this example let's assume that the overall net-net domestic cost is $65 per hour, on a simple rate basis it's hard to overlook the positive economics of a service provider rate of $45 per hour in China or $57 in India. A buying company can take on a substantial amount of buying risk for even $12 per hour over the range of a several thousand hour project engagement. If you work for a company who is considering a return to in-house servicing (known as retrosourcing) bear in mind that these economies will require reverse justification despite compelling circumstances.

Appropriate attention given to ways of reducing buying and estimation risks will minimize cost misconceptions and promote responsible risk control. Illustrated below are points worth considering.

1. There are efficiency and familiarization costs associated with first time outsource service transitioning. The buying organization can expect to realize less ROI during the initial engagement of outsource services. The speed of transitioning and acclimation will determine the realization of expected target ROI goals. A maximum of 10% transitioning impact can be expected (historically).
2. The influence of culture will contribute added cost. It is less a question of culture but the measures needed to overcome cultural challenges (language, customs, distance, time zone…). To illustrate, it is not uncommon that what took a chat in the hall may now requires a small written testimony to communicate with the provider of service. The good news is that this level of formality produces health benefits (better precision) and fulfills requirements directed through laws and regulation. Expected impact of 2% (historically).
3. Don't expect a smooth sail on the initial engagement or year of service offering. There are going to be things that neither of you expected or anticipated so remain adaptive and flexible. Close attention and vigilant communications will minimize the impact in this area. With proper attention no impact can be expected, if loosely attended to it can rise to as high as 5% (historically).

In summary, if you are anticipating a gross saving of 30% the preceding points will reduce savings to a net of 18% in the initial orientation period and return ROI to a healthy level in the order of 23-25%.

Although there are few public studies or metrics available one can expect a gross savings in the area of 20-40%, and when considering the initial transitioning costs of 10-20% (a nearly break even situation) at the onset of the relationship. However, don't be scared away. This remains short term, controllable and offers added flexible service delivery that would not have been possible using conventional domestic resources. Over the run of a service project gains of 15 to 20% are common (a figure much better than one could realize from traditional investment choices). Even small one time service engagements will have lower cost effects with yields in line with longer term engagements. This is caused to a large part by an appropriate lower level of formalization that would be attributable to longer term and larger service projects.

Purely based of economics outsourcing in both the short and long term can produce substantial gains and a positive ROI for companies.

The Next Steps

Let's assume we have dealt with the leading risks involving engagement and estimation management. Our next hurdle is to determine what region to seek qualified suppliers from. Some consider this simply a matter of choosing a popular destination and start canvassing the thousands of potential suppliers. This approach, while quick, may totally miss the optimized economic objectives that have been set. In keeping with our theme of 'positive economics', a good place to start is where we can get the most for our money. Currency valuations, trends and current status help to influence what region might make the most sense to seek service providers. For example, if we look at how much we can get for our dollar ($), euro (€) or pound (£) this would help narrow the group of service providers. At present;

What does this chart tell you? Are there clear choices or does it raise further questions for added consideration? Does the stability of the currency influence your decision or are you willing to take some risks and hedge your aggressiveness with a dabbling of monetary future speculations. These represent a small fraction of the questions that are worth asking. Obviously our interest is to optimize the economies of the outsourcing engagement and while we could overlook such matters we might be facing the unfortunate decline in monetary value. This would jeopardize and potentially put at risk our ROI. Some additional factors for consideration include,

- Are suppliers requested to quote in native currency or in local buyer currency? Keep in mind that quotes in buyer currency may be padded for currency fluctuations. Given present economic conditions, deflated USD ($)s will be more expensive in the future whereas supplier currencies are poised for a reduction. Whoever controls the currency will control the risk.
- For longer term service delivery engagements buyers may wish to consider the economics of currency futures. While there are investment risks there are opportunities to develop gains that can be used for a variety of other purposes (e.g. funding estimation errors or new feature additions). The previous chart does not reflect whether the respective currencies are at the bottom or top of trading. However, iwhat is up will likely go down, and what is down will go up. Obviously there is a lot more to these topics (*) and one should seek the advice and counsel of your corporate investment department when formulating your outsourcing strategy.

	USD ($)	Euro (€)	GBP (£)
India (INR)	49.18	63.7129	67.9751
China (CNY)	6.842	8.8638	9.4568
Russia (RUB)	32.696	42.3578	45.1914
Brazil (BRL)	2.3458	3.039	3.2423

22 January 2009 - Forex

- Utilization of currency rate information can provide guidance on whether to utilize fixed price or time-and-materials outsource engagements. High volatility or economic uncertainties are suitable for fix price engagements. Low volatility with reasonable economic expectations offer sufficient value through a time and materials relationship. In both cases aggressive buyer (and supplier) project management will insure mutual engagement benefits.
- How do the costs match for specific service project roles? Even though a specific country may offer favorable currency exchange rates (and it's relatively stable) consideration must be given to cost per role.
- All roles need to be reviewed collectively. By way of example, if there are two roles that have a relatively higher rate, but the remaining 14 roles have a lower rate for a longer consumption period then this choice might be more economically beneficial. Usually in these circumstances additional questions about qualifications, tenure, turnover and language skills may be explored.

Are We Done Yet?

In terms of outsourcing economics, the answer is yes (almost). But in terms of realizing the value the fun is just beginning. Despite proper care and due diligence, estimation adjustments and risk reduction economic success will hinge on aggressive and coordinated project management. Utilizing the fundamentals of sound project management, such as the Project Management Body of Knowledge (PMBOK by the Project Management Institute (PMI www.PMI.org)) ,specialize outsourcing related project management topics, focus, and direction exists. This include unique attention to,

- Global coordination,
- Communication systems, vehicles and formalization,
- Pre-engagement synchronization,
- Advanced coordinated metrics, and
- Shared service delivery.

A targeted body of knowledge exists specifically for global outsourcing project management and can be obtained at www.Int-IOM.org (Outsourcing Management Body of Knowledge – OMBOK). Simply put, the economics visions formed as a part of service project definition can only become a reality through purposeful joint commitment by buyers and suppliers. Your governance responsibilities continue no matter whether you continue to provide direct services or utilize the resources of an independent supplier. Continued oversight, care and attention remain and will last for as long as services are provided to stakeholders in your company.

Conclusion

Can outsourcing yield a positive ROI?	*Yes*
Are the yields as high as we expected?	*Probably Not*
Should we expect some added costs?	*Expect some*
Will we need to realign our approaches?	*Definitely*
Should we expect that some changes will need to be made?	*Probably*

The good news is that there is an abundance of case studies, articles, and resources that can be put to work to overcome risk, guide in the realignment of processes and methods, and assist in the delivery of high yielding risk overt outsourcing. As a buyer, or as an experience veteran of outsourcing, the opportunity to reinvent/rediscover the economics has come at a time when success may mean survival. Economics lead the score card of reasons behind choosing outsourcing as an operational alternative. However a close and critical examination of other service provider attributes will be necessary. Question of supplier viability, regional stability, intellectual property rights (IPR), delivery capability and organization synergy are but a few of the other factors that will influence your economic realization.

Biography

Jerry is Founder and Chairman Emeritus for the International Institute for Outsource Management (IIOM). He has provided support and guidance to some of the leading global companies in over 70 countries. The focus of IIOM is to serve the growing and varied needs of the global outsource service provider while alleviating many of the issues expressed by global buyers. Jerry lead the development of the Outsourcing Management Body of Knowledge development, and established the Global Star Certification (GSC™) a model focused specifically on outsource supplier viability. He presently serves as a Senior Global Advisor for the Outsourcing Institute, Bocosoft, Tianjin Software Testing Center and Beijing Association of Software Sourcing. Question and feedback can be directed to: jdurant@Int-IOM.org or IIOMDurant@gmail.com .

Outsourcing And Unit Testing

by Jean Joskowicz

Since specialists have been wondering about the best way to develop major software projects, they at least agreed on the fact that the size of their teams was usually growing progressively according to the advancement level of the project, until it reaches a maximum value, then a kind of plateau phase, followed by a decrease, giving to the manpower resources curve the remarkable shape of a sperm whale.

Whatever the name given to this step (programming, coding, other..), in the realization phase, where this plateau stage is noticed, it appeared that at that time and during this particular step, a maximum number of IT specialists was requested, let them be qualified as programmers, systems analysts, even coders. And project managers rather quickly imagined that it would be possible to sub-contract all or part of all of the corresponding tasks to external and even foreign companies(1), in outsourcing or off-shore modes.

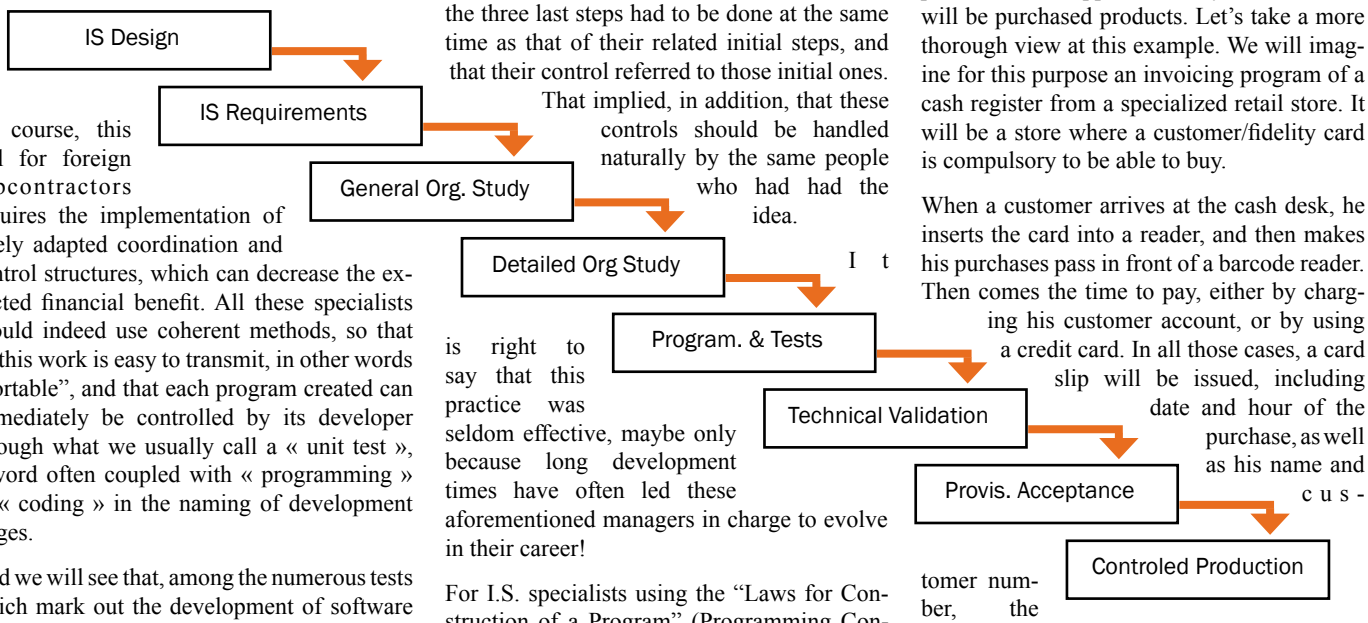
Of course, this call for foreign subcontractors requires the implementation of finely adapted coordination and control structures, which can decrease the expected financial benefit. All these specialists should indeed use coherent methods, so that all this work is easy to transmit, in other words "portable", and that each program created can immediately be controlled by its developer through what we usually call a « unit test », a word often coupled with « programming » or « coding » in the naming of development stages.

And we will see that, among the numerous tests which mark out the development of software projects and their slow migration towards the state of a successful application, these "unit tests " are taking a central place in the project development, as we will now see.

Indeed, when thirty years ago quality experts decided to fold up the "Waterfall Cycle" of project development to get a "V Cycle" (without actually making it any more cyclic from a strict point of view), they were guided by the idea to associate at the same level the execution of a test type and its preparation.

As a result, the lower point of this pseudo-cycle became the programming phase, including coding and unit testing of the program.

The doubled relationships between « Information System Requirements » and « Production under control » on the one hand, « General Organic Study » and «Provisional Acceptance" on the other hand, and finally "Detailed Organic Study" and "Technical Validation" were highlighted to make obvious that the design of the three last steps had to be done at the same time as that of their related initial steps, and that their control referred to those initial ones. That implied, in addition, that these controls should be handled naturally by the same people who had had the idea.



Process 1 « Waterfall Cycle»

It is right to say that this practice was seldom effective, maybe only because long development times have often led these aforementioned managers in charge to evolve in their career!

For I.S. specialists using the "Laws for Construction of a Program" (Programming Construction Laws) developed by Jean-Dominique Warnier, executing this kind of tests should not be a problem.

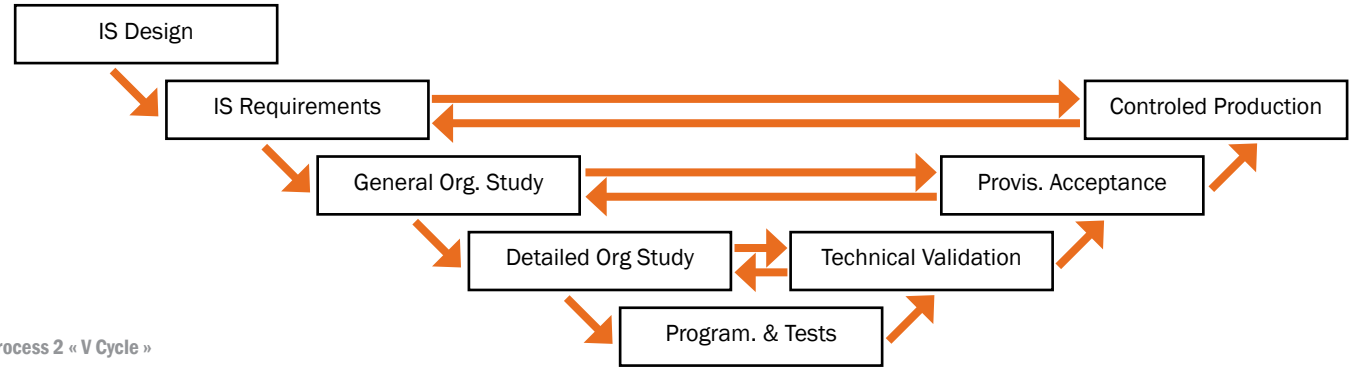
Thanks to the LCP methodology, it is easy to know which tests should be executed to validate a program.

In fact, every program breaks up into alternative or repetitive structures, which highlight the conditions to realize. In theory, each branch of an alternative structure should be at least executed once, and each repetitive branch twice, except for some specific cases.

Another principle is that each output data should be elaborated in a sequence executed as many times as the output should occur.

For instance, in an invoice intended for a customer, the customer's name should appear once per customer. But each purchased product should appear as many times as there will be purchased products. Let's take a more thorough view at this example. We will imagine for this purpose an invoicing program of a cash register from a specialized retail store. It will be a store where a customer/fidelity card is compulsory to be able to buy.

When a customer arrives at the cash desk, he inserts the card into a reader, and then makes his purchases pass in front of a barcode reader. Then comes the time to pay, either by charging his customer account, or by using a credit card. In all those cases, a card slip will be issued, including date and hour of the purchase, as well as his name and customer number, the



Process 2 « V Cycle »

name and value of the purchased goods, the invoice total, and the payment method used.

We will not detail the whole LCP[1] process, but we will directly show the flow chart which relates to this context, and the results of this.

We can note that there is first a repetitive structure on the product process, then, at the time of choosing the payment method, there is an alternative.

The program comprises six logical sequences and two tests. The first one could be named "new product occurrence?" and the second one "choice of payment by card?". In the sequence of "Begin Customer", we will input the date and hour of passage of the customer (provided by the system), then the input of the customer's reference (through the card reader) will enable to access to the data necessary to create a customer invoice (name, address, …). The totalizer of the customer purchases will be initialized in this sequence.

The sequence « Product Process » will correspond to the identification and valuation of each product, which will be cumulated in the totalizer.

The total will be obtained in the sequence "Intermediate". The chosen method of payment will be naturally notified in the corresponding sequence. The complete invoice will be finally issued at "Customer End".

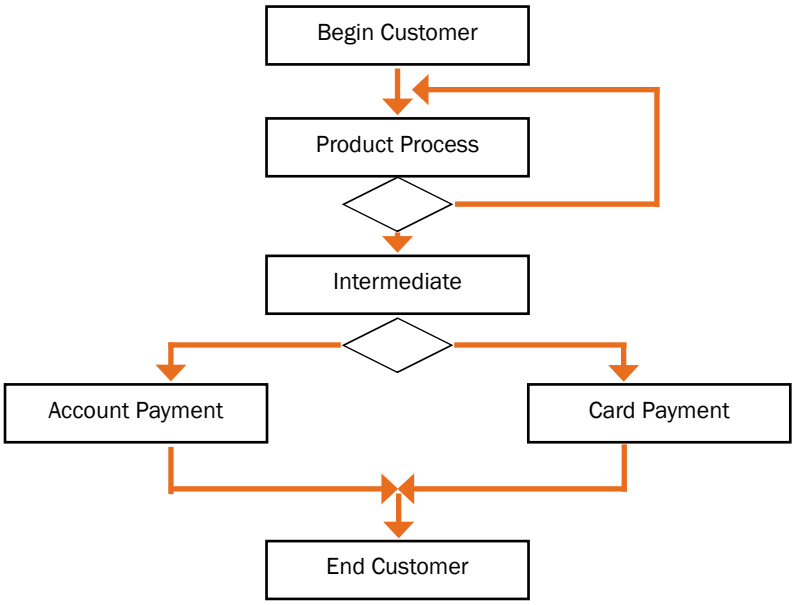
The test data enabling to validate this program will require two customers, one paying by credit card, and the other by charging his account. One will have a single product, while the other will have at least two. In this way, all branches will be tested.

In this version, the program is not planned for a customer who would not make any purchase, for example by canceling the bought products. Such a cancellation of product, a frequent occurence in real passages at cash deck, is not envisaged by this program either. That could be the object of a modification of the program.

Which conclusions can we draw from this quick study? Quite simply, the fact that testing can be organized and prepared independently from the programming language! The test logic that we have highlighted shows that the Warnier process corresponds really to a programming logic, which its inventor called "data processing logic" ("logique informatique" in the original French language).

And if we want to evolve from a simplistic program to a more realistic program, what will we see? Let's try!

Going from a single customer to a group of customers, is achieved quite simply by integrating the "Customer Process" into a repetitive loop, which then allows us to execute our tests cases during the same passage.



Furthermore, if we want to get daily summaries, for instance, by cash register, assuming that no customer will arrive at this cash register, it will be enough to introduce an initial alternative, creating what Warnier called an "alternative group".

We are free to make this example even more complex, for instance by introducing the concept of "private customer" or of " professional customer" , through a special code of the customer fidelity card, to be tested at the beginning of the customer process, allowing thus to benefit from specific advantages, bonus, discounts, and so on.

What we can conclude from all this is that all testing should be first specified in the most inner parts of program (deepest "levels" of LCP structures), before going back to the most external levels.

1 Let's remember that it is necessary to first structure the Logical File of Outputs (LFO), then the Logical File of Inputs (LFI),in order to proceed to their mutual validation, and then to deduce the structure of the LFI program by applying the Laws for Building a Program.

Biography

Engineer and graduated as a DEA in Physics, M. Jean Joskowicz began his career in scientific research, before choosing business EDP.
A specialist in structured methods of programming, conception and project management, he became involved as a teacher in national or international organisations.
As senior consultant in various consultancy and services forums, he managed or rescued large projects in France and abroad.
He is President of AFISI and international consultant, and as such he participates in international lectures and conferences and participates as trainer at Test&Planet.

Test your reality with Cognizant.
Compete with confidence.

Whether its supplementing your existing program or providing an end-to-end solution, whether it's a long-term testing roadmap or detailed process improvements, you can count Cognizant's consistency, discipline, and operational efficiency.

There are many advantages to working with Cognizant. We'll help you look beyond the obvious, provide you with independent suggestions for quality improvements that will mitigate risks and improve software performance

With Cognizant, you get certified continuous business-readiness, lower costs, and accelerated results unmatched in the industry.

Please feel free to reach us at
testing.services@cognizant.com

www.cognizant.com\testing

Cognizant | Testing Services
Passion for building stronger businesses

World Headquarters
Cognizant Technology Solutions
500 Frank W. Burr Boulevard
Teaneck, NJ 07666
Phone: +1 201 801 0233, Fax: +1 201 801 0243
Toll Free: +1 888 937 3277
Email: inquiry@cognizant.com

A desire for cost cutting is not the best starting point for offshoring

by Rik Marselis & Holmer Vonk

Due to the economic crisis an enormous pressure to reduce costs exists.

Therefore companies easily consider offshoring their activities, for example testing, without paying attention to the necessary preconditions. Without satisfying several preconditions, offshoring usually does not lead to lower costs (on the contrary…).

We have developed an approach that has proven to bring the expected results at reduced costs. However this approach does not start by aiming at lower costs. It starts with determining the business goal. The approach to testing is tuned to this goal. And it focuses on testing in a smarter way. After some specific preconditions have been fulfilled, a mix of outsourcing and offshoring may be the perfect way towards that business goal and reduced costs can be one of the consequences.

But without a smart setup of testing, disappointment usually follows on outsourcing and offshoring.

Smarter testing leads to lower costs at equal results

To optimize testing the test specification and execution activities have to be made independent of people, time and place. After creating this independence the test manager can decide on a detailed level, which activity should be done at what location. This decision is important since we have learned that a "one-size-fits-all" solution never fits all. So a differentiated approach will lead to a better result and lower costs, not only for testing as such but also for the business as a whole. On the other hand this differentiated approach requires a well-thought-out organization. In the traditional approach the creativity of people is frequently needed to keep the process going. But creative solutions often are not the cheapest solutions.

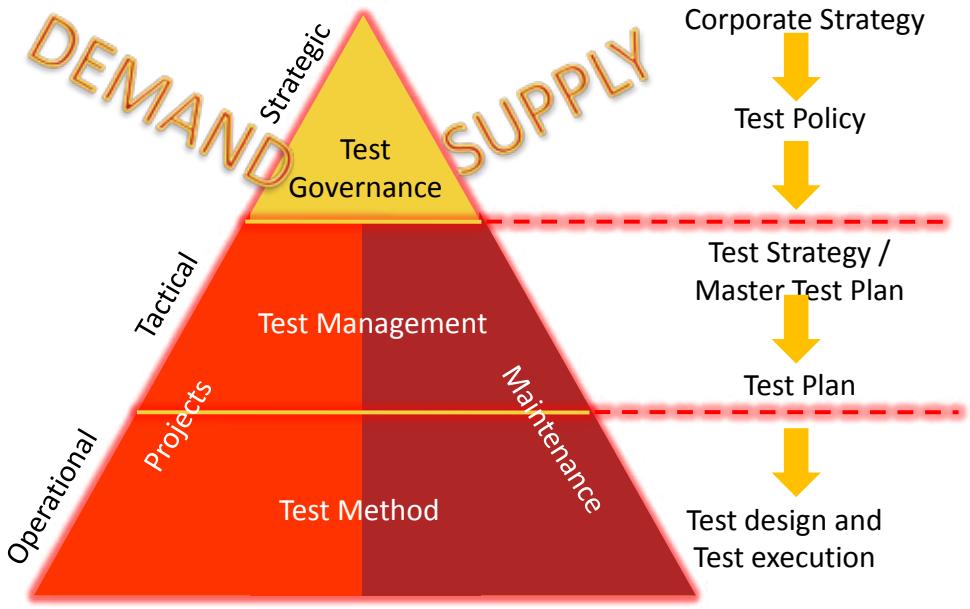
So the first step to smarter testing will be to "organize".

Alignment with the corporate strategy enables a good organization

It is common knowledge that the corporate strategy has to be translated into policies in order to make an organization work towards the business goals. Too often we still find that this is not done for testing. A test policy however is the primary aid to focus testing activities towards one common goal. The benefit of a policy is that it has a long-term perspective. The desire to outsource or offshore must fit this perspective. Traditionally offshoring cannot be done for a short period of time. Performing work offshore usually takes quite some organizing, so the test policy first has to be translated into the organizational needs. This creates a very important starting point for the decision on where the testing activities will be done. Then, again based on the corporate strategy, decisions can be made on what parts of work will be outsourced and/or offshored and what work will remain on-site. This way the corporate strategy is also the point of reference for deciding upon the actions to face the current economic crisis.

A client remains accountable when outsourcing / offshoring

One of the common misunderstandings with respect to outsourcing and offshoring, is that all responsibilities are handed over to the provider. However, the client still remains ac-



countable towards the business goal. So the client must be sure that governance is well organized. This calls for "testing governance". The commonly referenced demand-supply model needs governance, IT-governance and of course testing governance, to coordinate between the business representatives at the demand side and the provider representatives at the supply side, thus taking care that all activities are mutually aligned and also aligned with the corporate strategy. Testers often think in terms of projects or maintenance. Testing Governance takes care of activities which are aligned for both projects and maintenance, since testing is about ensuring the added business value in business processes and not just about measuring quality in one stage of an IT-system's lifecycle.

Now that we have learned the importance of organizing testing well, the next step to smarter testing is to "communicate".

Communication of goals, assignments and results is essential

When we hear people talking about outsourcing and offshoring it is often about obstacles. Language, culture and traveling are often mentioned. These obstacles can be overcome by paying attention to communication. When starting to work independent of people, time and place a first step is to enable people to do so. Independence of place implies that the language must be common for all places involved. So if different languages are used, make an agreement on what common language will be used and never translate things more than once. When personal communication using a telephone is difficult consider using instant messaging since reading another language is often easier than hearing it.

Culture basically is not a problem at all. Everywhere you'll find more and less motivated people. But there is no culture that prevents people to cooperate. However people do tend not to understand each other. That is not a cultural issue; it results from different frames of reference. For example because the parties involved have totally different maturity levels. CMMi level 1 people and CMMi level 5 people simply cannot work on one project.

So communication is basically the problem. Regular contacts (at the start in person, later using telecommunications) help improve working together. When the teams are used to working together hardly any traveling is needed but, more important, you will see that the mutual prejudices of remote teams towards each other will disappear. Thus, taking care of teams that are used to work on the same type of testing activities enable working independent of people, time and place.

A standard way of working improves cooperation

Essential to proper communication is that everybody involved knows what to do, what to use and what to expect. So clear agreements on the processes and products must be made. This will be supported by using common tools, templates and checklists. Especially when the parties involved have all participated in setting up this way of working to prevent essential unworkable situations. Configuration management will ensure that the test basis, test object and testware are always aligned. Knowledge management will ensure that a tester that is acquainted with the test method does not need subject matter expertise to be able to do the test design and execute the tests. Instruments used are documentation, wiki's and a person as functional oracle. A learning process that includes determining root causes of incidents with the information from the help desk prevents defects and other problems from returning.

Taking care of organization and communication prepares for the final step of smarter testing which is to "industrialize".

Well-defined work-packages enable flexibility in people, time and place

The client knows what results have to be reached and when. Creating those results is a certain amount of work with specific deliverables as defined in the assignment from client to provider. For each deliverable the test manager defines work-packages with the relevant part of the test basis and the test object and the products to be created, such as test cases, test data and the test summary report. The test manager decides which work-package is picked up at what moment and by what team. Since the work-packages are defined in a standardized way there is no precondition to who should work on such a work-package, therefore every work-package can be dispatched to an available team member no matter where he is located and what time-zone he is in. A common set of criteria is used to determine per work-package to which team(s) it could be dispatched.

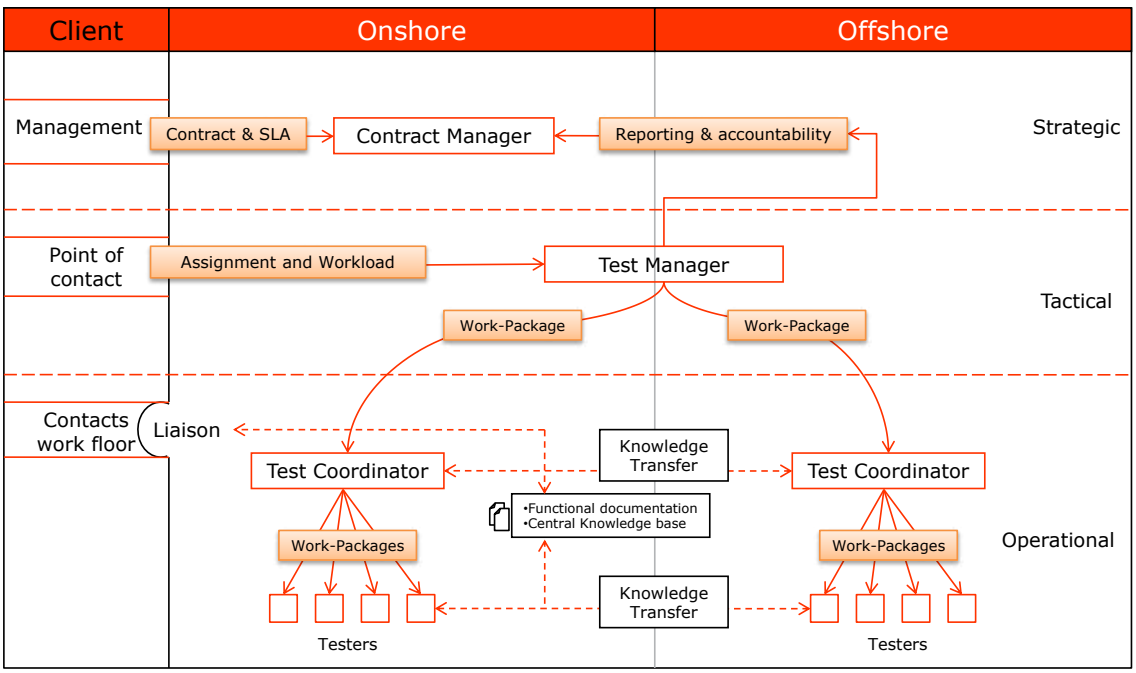
The true advantage of using work-packages is that when somewhere in the process an activity is dependent on another activity which causes a waiting time, during that waiting period the people don't run idle but can pick up another work-package of the same or another assignment. This is a true enabler of reduced costs!

The client does not need to buy test tools

A key issue to industrializing is not to do superfluous work. Many activities can be automated. For testing activities like test design and test automation this is the same. But the initial costs and start-up activities often are a hurdle that's too high. A service provider in testing nowadays has licenses for a test-tool-suite of which the testers have knowledge and experience. This is part of the knowledge management approach. So in an industrialized approach the client does not need to worry about tooling, the testing is done in an efficient way at optimal costs.

Customer intimacy is crucial to beneficial outsourcing

When a client wants to take advantage of possible benefits of outsourcing a last obstacle is often the uncertainty of who is going to help him when he is in doubt. To solve this we have created the "Testing GateWay" principle. This ensures proper communication between the customer on the one hand and the test team on the other hand. For the client there's no difference whether the test team is onsite, onshore or offshore.



The Testing GateWay takes care of all three levels of communication.

At the highest level the clients management establishes a contract with the contract manager. At tactical level the point of contact on the client side discusses the assignments with the test manager. And on operational level the subject matter experts have their liaison who is the communications channel towards the test teams. Now that each level of representatives at the client-side has its own single-point-of-contact the normal disadvantage of needing extra test management effort when outsourcing and offshoring is diminished as well as the usual startup times.

In this Testing GateWay the quality of the test basis is determined and thus in a very early stage the obtainable level of test results (which have a direct relationship to the quality of the documentation) can be established and communicated to the client.

STaaS facilitates outsourcing in situations that wouldn't be feasible before

Organizing testing as described above, using key-points like work-packages, standardized tooling and independence of people, time and place, has made it possible for us to create Software Testing as a Service (STaaS). This service makes outsourcing attainable even for small assignments, like a one-week functional test of a small website or the usability check of a back-office application. The standardized approach of STaaS even enables us to make a quality assessment of a test object for which no proper test basis exists. Branch specific test sets and generic checklists are the key to this level of test services. Regression tests often are relatively small and well defined tests that can easily be fitted in proper work packages. Therefore regression tests are a perfect example of tests that can be done through this service concept.

A mixture of offshore and onshore activities works best

Efficient and effective testing does not call for either offshoring or onshoring, it requires a balanced total of testing activities done by the people that can do it best given the circumstances, at a time suited to the client's needs and at a place where the testing can be done as quickly and cost effective as needed.

Conclusion: Smarter testing is the starting point if you desire cost cutting

So if you feel a desire to reduce costs keep in mind that outsourcing and offshoring in itself are not automatically the solution. The keywords are "organize", "communicate" and "industrialize". And after taking care that the work has been made independent of people, time and place, by introducing the concept of work packages, you can implement your perfect mix towards reaching the business goals at acceptable cost. The Testing GateWay principle establishes the essential single-points-of-contact which, in combination with Software Testing as a Service, provides a proven example this approach reaches the same or better results from testing at reduced costs.

Bibliography

TMap® Next for result-driven testing, T. Koomen, L. van der Aalst, B. Broekman and M. Vroon (2006)

TMap NEXT® Business Driven Test Management, L. van der Aalst, R. Baarda, E. Roodenrijs, J. Vink, and B. Visser, (2008)

Samenhang vraagt om slagvaardige regie (Coherence calls for sharp governance) R. Linkers, R. Koornneef and H. van Mastrigt, article in RealIT, june 2008

TestGrip, gaining control on IT quality and processes through test policy and test organisation, R. Marselis, I. Pinkster, J. van Rooyen and C. Schotanus (2007)

Biography

Rik Marselis is a test strategist within Sogeti in the Netherlands and one of the designers of the STaaS concept. He has over 28 years of experience in IT of which about 14 years in quality assurance and software testing. Rik has presented on testing subjects like test policy at various conferences, including Testing & Finance 2008. Also he wrote numerous articles and contributed to various books on testing. As a trainer Rik works with groups in a wide variety of training courses on software testing. He is the treasurer of the Belgium and Netherlands Testing Qualifications Board (one of the 42 ISTQB member boards) and he is an active member of TestNet, the Netherlands special interest group in software testing.

Holmer Vonk is process manager for offshored test projects run in cooperation between Sogeti Netherlands and Sogeti India. Holmer has 10 years experience in IT, which started with education of software applications to businesses. The last 6 years Holmer built up knowledge and experience in migrations of Operating systems, testing, test management, CMMi assessments, process development and offshoring. Recently Holmer implemented the Testing GateWay principle, as described in this article, within Sogeti.

Mid-Term Health Check

Where Do You Stand at the Half-Time Mark?

by Nishant Pandey

A wide range of status trackers and progress indicator tools are available to the test leads and managers, but are these sufficient in themselves to help pull back a test project gone wrong? A "Mid-Term Health Check" meeting can serve as a useful tool. This project review meeting can serve as a stitch in time.

A mid-term health check exercise can help the test project stakeholders to gauge the actual status once the execution schedule has reached the half way mark. The article focuses on the need and importance of a formal "Mid-term health check" exercise.

Mid-Term Health Check:

A well planned and efficiently conducted mid-term health check meeting can help answer many questions for the project stakeholders, like (but not limited to):

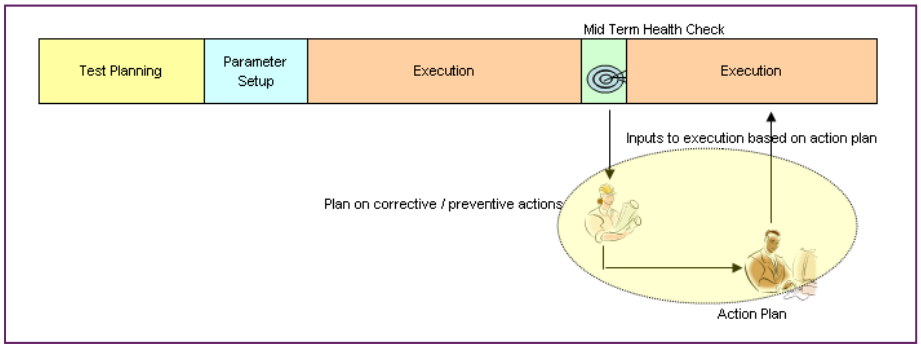
- Will the test execution be completed on time if the progress continues at the same pace?
- Are the resources allocated to the test execution too many or too few?
- What is preventing targeted completion to occur on the expected date?
- Has the retest effort (arising due to bugs in code) been estimated correctly?
- Are there any actions that can be taken now to ensure the test project's success?

A typical mid-term health check meeting should ideally be conducted when the execution schedule reaches the mid-term mark. The half way time is considered appropriate, as generally by this time all shakedown activities and initial glitches have been removed and not only has sufficient time elapsed to establish execution trends, but there is also sufficient time left to take corrective and preventive actions.

The agenda for the mid-term health check meeting could contain the following items:

- Execution trends and percentage of completion to date
- Execution progress predictions
- Deviations from expected progress and reasons for the same
- Open issues and risk items
- Defect trends and retest estimates
- Action owners & management support

The following case study provides an example of discussion items at a typical meeting.



Case Study – Mid-Term Health Check Exercise:

Project "XYZ" that involved testing of four key modules of a financial services product was started on time and the test plans were created as per the company standards and policies. The test execution exercise was started as planned and the test team started off well, completing more than the expected test cases for each month. However, it was found that after 2 months of execution the test case execution was not progressing as fast as expected, soon the team was lagging behind the weekly targets. The Test Manager called the mid-term health check meeting and here are the findings:

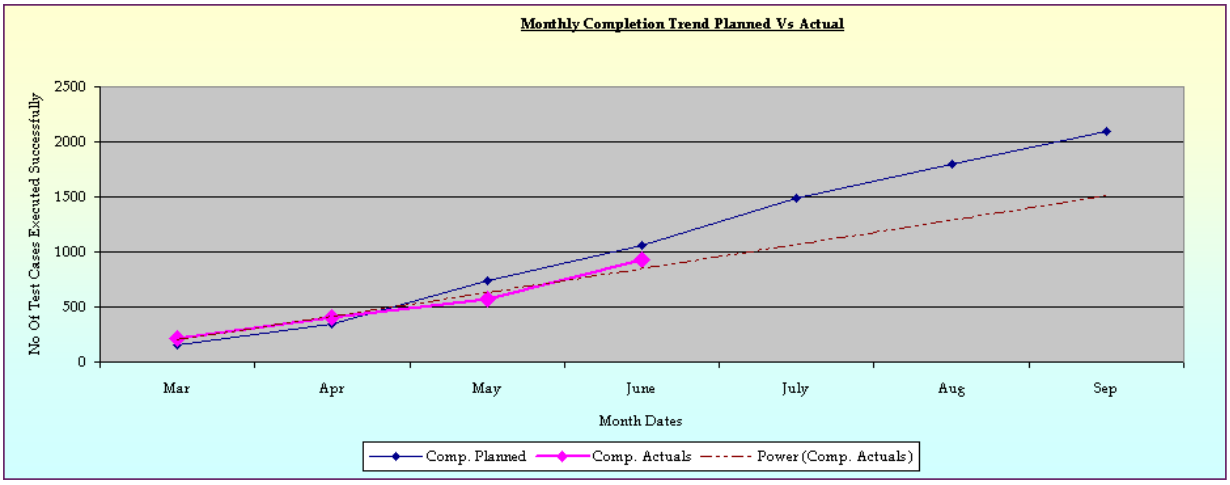
Execution Trends and Completion Rate:

The monthly completion of test case execution for the XYZ project was ahead target for the months of March, April and May. However, the test team started to lag on this front as of May onwards.

The project execution trends were as shown in fig. "Monthly Completion Trend Planned Vs Actual".

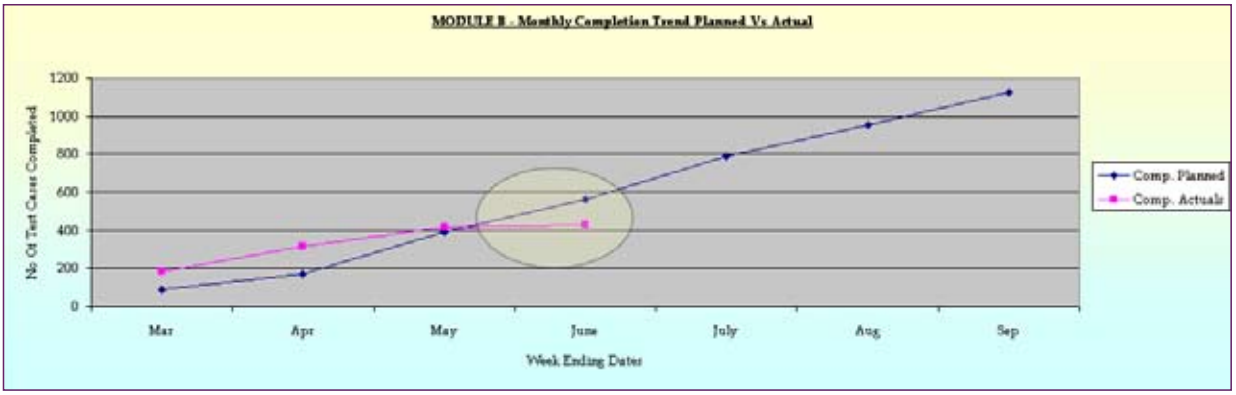
The trend analysis revealed that though the team had started off well, execution was behind target and if things continued this way, the project test execution would be short by around 500 test cases by the targeted completion date in September. To get down to the exact reason for the lag, the test manager drilled down into execution progress trends for individual modules. It was found that, out of the 4 modules under test, 3 were ahead of the targeted completion, and the lack of progress on one of the modules (Module B) was the root cause.

The execution trend for Module B was found



to be as shown below:

It was found that lack of progress on module B during May to June was affecting the execution totals. Inputs from the module test lead led to the knowledge of the fact that though the test team was working hard towards meeting the targets, the high number of complex bugs and repeated failure of retest fixes were the main reasons. The test lead helped identify the list of bugs in priority order that can unblock test cases for execution once they have been resolved.



Defect Trends and Retest Estimations

The execution trend analysis pointed towards unusually "buggy" code as one of the probable reasons for the variations between actual and targeted completion. It was found that the test team had found 800 defects by June. Based on trend prediction it was estimated that the team will end up finding around 1200 bugs by September. This was higher than the estimated bug traffic. Discussions with test team members at the mid-term meeting helped reveal that, based on organization baselines and past data, the test project retest estimations were done to cater for 900 bugs and related retests. The unusually buggy nature of the code was leading to test resources being utilized in retest-related activities and thus impacting progress of execution.

Action Plan Creation

Based on the discussions and findings of the mid-term health check meeting, the test team identified the following action items:

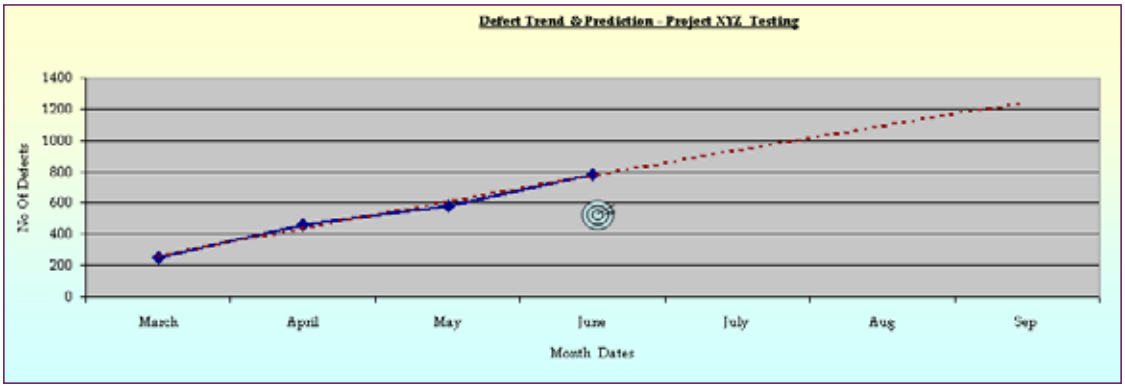
- Prioritization of defect fixing and increased focus from Development Managers
- Allocation of extra resources for retesting the open defects
- Increased focus on Module B and updates to risk plan
- Commitment from management for monthly reviews

Results

The mid-term health check meeting helped serve as a powerful communication tool for the test manager to put across his team's concerns and bottle necks that were causing a dip in the execution status. Timely prioritization and corrective actions by project stakeholder helped ensure that the project was completed on time and with the required depth of testing activities.

Best Practices & Pitfalls to Avoid:

- **Prepare and communicate ahead of time** – your test leads will not only require the exact details of the metrics you need, but will also need time to get you the data.
- **You will need support** – Involve all important stakeholders. Identify action owners and get management to buy in from the concerned departments.
- **Make it official** – Once the benefits of such meetings have been achieved and the importance realized by the team, include the mid-term health check in your test methodology.



- **The first health check** – Make sure you have the processes setup for collection of the required metric data that you plan to analyze. If a metric data collecting mechanism is not a part of your team's test processes – spend some time to start at this front before the project execution starts.
- **Use the metrics wisely!**

Biography

I have 7 years' experience that revolves around QA, Business Analysis, Test Management & Project Management. Armed with the advanced level certificate in Test Management (CTAL-TM) and the PMP (PMI) credential, I am involved in test management and test process improvement in Capgemini FSSBU. I have worked on Credit Cards and Telecom Domains. Currently I live in Orlando where I am working as Test Manager for a prestigious client of Capgemini FSSBU. My first paper on Software Testing was published in proceedings of the National Conference on Software Engineering (NCSOFT - India). Since then I have been working on pinning down some of my experiences and will be glad to share these with others.

Us and them: when cultures collide

by Jos Paeshuyse

Outsourcing: the transfer of responsibilities, goods and tasks to third parties.

The latest trend in software building and testing is outsourcing and off shoring of activities to countries with low costs and wages. Companies establish partnerships with companies in these low wages countries. Their goal is reduction of costs, increasing profits and continuity.

The world is getting smaller. 30 Years ago different cultures were an attraction for people who went on holidays to other countries. We were treated to interesting stories from travelers who told of strange people with strange habits and different ideas. But fortunately we could return home, to business as usual. Back to our trusted environment. Difference in cultures in business environments is often the cause of trouble and far less a cause of joy.

Times change and nowadays our society consists of more than just our own culture. It is a motley collection of nationalities and cultures, of people with different habits and ways of thinking and acting. Today's problems often submerge from misunderstanding and incomprehension.

This new trend leaves us with new challenges. First of all there is the business side of the partnership. Contracts are signed and service level agreements are being negotiated. Let the show begin.

After the first euphoria has disappeared the first problems emerge. Frequently they are unexpected and come from a non-business side: the difference of cultures. These problems are hard to capture in a contract or SLA. It has to do with people and their peculiarities. Often heard problems are: "They do not do what we expected them to do", "They say YES to anything, but they do NO", "They don't get us", etc. Programmers do not program what they should according to the functional design and testers do not test according to our standards. In other words: they do not deliver what they should.

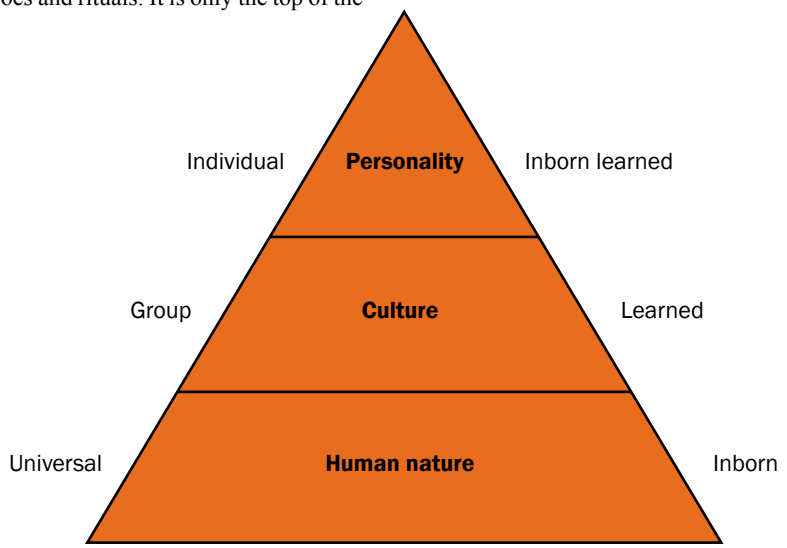
In many cases these problems can be captured as "Cultural differences". Values and behavior are different, communication is different. Learning a common language (in most cases English) is only a small part of the solution.

Culture is not in our genes, but is something we learn in our childhood from our parents, at school and in society. Therefore: cultures can vary a lot and so do the values and behavior of people. A judgment of the other culture is often based on incomprehension.

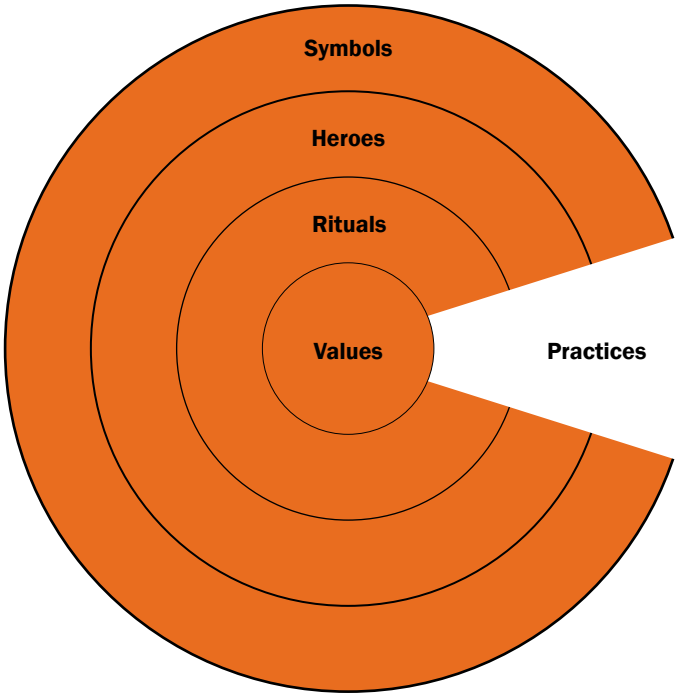
The picture above shows that culture is specific for a group of people and shines a light on human behavior.

What we see of cultures are the practices: symbols, heroes and rituals. It is only the top of the iceberg. Below the surface is the hidden, but persistent part: the values of a culture. They are not always visible at first sight and are very hard to change. They are programmed into us as we grow up.

In some cultures the individual leads and, in others it's the group. Some cultures are rigid and focused on the long term while others are flexible and willing to take risks. In some cultures the word NO hardly exists (they say YES, meaning "I hear what you say") while others are direct and provide immediate feedback. Trading with some countries means that you have to ask explicitly for a lot of things because in their culture a lot of information is included in the context of the message they give you. Documents you get from these countries can be very minimal. This can be killing for cultures that are used to putting everything on paper. Another issue can be the role of men and women in a culture. Is a female tester allowed to report a defect (in other words: be



3 dimensions of mental programming (source: Hofstede and Hofstede. 2005:20)



Cultural expressions (source: Hofstede and Hofstede. 2005:22)

critical on the work) of a male programmer according to her culture? In some cultures the loss of face is so important that testers hate to log defects they find. A defect tells the programmer that he did not do his work the right way and what if the defect was unjust? So logging a defect is a cultural problem, one way or the other.

Other things that can influence your communication with other cultures are religion and the distinction of the boss: are employees allowed to do anything without consulting their boss and. if not, does the employee say "No, I can not" or does he say "Yes" and does nothing.

We should not have the idea that cultures can change over night because they often go back ages. And why should they change? Working with other cultures should stimulate our creativity by approaching problems in a way that we haven't thought of ourselves. So, what we are left with is the growth of awareness. Awareness of our own culture and peculiarities and awareness of the culture and peculiarities of the other culture so we can prepare ourselves. This way we can reduce or even prevent problems, reduce costs and increase the chances of success.

The recognition of cultural differences can be learned. A good training can help you to avoid problems as mentioned before. The basic idea of a training in Intercultural Communication is to get you from the stage of "not knowing and not liking" to the stage of "recognition and acceptance" and further to the stage of "integration and transformation". Awareness is the magic word; awareness of your own culture and awareness of other cultures. Through this you can interact with the other culture more easily. You will be able to predict the behavior of the other culture and deal with it properly.

Software development is teamwork. Having a multicultural team is an extra challenge for every manager. Software testing, as a part of the development process, is no exception to this rule. With that a good Intercultural Communication training is indispensable for a well greased multicultural team. It is indispensable for the success of outsourcing and off shoring. Therefore it is a profitable investment for your company. There is more to it that good contracts and service level agreements.

Biography

Jos Paeshuyse was born in 1956 in the Netherlands. He started his ICT career in 1985. Since then he operated in various roles. During the last 12 years of experience as test manager he was involved in projects that outsourced activities to countries with low wages. In this period he became convinced that is was hardly possible to change the culture of people, but the way to deal with other cultures was the growth of awareness. He is still working as a self employed test manager and owns a company that organizes workshops for Intercultural Communication for companies who are getting involved with different cultures. His main interest is the Indian culture.

Limited places

TMMi Workshop

one day workshop by
Geoff Thompson and Brian Wells

Wednesday 20th May, The Royal Automobile Club, 89 Pall Mall, London, SW1Y 5HS
9.30am – 5.00pm

In a recent survey of CEOs and their strategies to contain costs in the coming year, improving IT processes came top of the list. This not only demonstrates that CEOs know a lot more about the workings of IT than they did in the past, but also that they are aware of the tangible benefits associated with improving IT Process efficiency.

The impact of process improvements throughout the SDLC can have a major impact not only on IT's capabilities to deliver, but also the knock on affect to the business. (as outlined in the table below)

Business	IT
Competitive Advantage	Reduced Time to Market
Cost Reduction	Increased Productivity
Risk Mitigation	Risk Management
Increased Business Efficiency	Improved Quality of Deliverables
ROI on IT	Measureable Improvements

Over the decades there have been many process improvement models, including SPICE, CMM and CMM(I). These models focus on helping organisations improve their overall software engineering process capabilities, but don't cover testing and software quality processes in sufficient detail. This is where the Testing Maturity Model integrated (TMMi) can help.

Unlike other Test process improvement methods that are attributed to individual consultants or organisations TMMi is the standard industry method to assess and improve testing and software quality related processes.

TMMi brings together a prioritised and coherent strategy to make the practice of testing, a validation of quality and not just a way of finding defects.

Experimentus has obtained wide recognition for the work it has been doing with the Testing Maturity Model integrated and has helped many organisations clearly understand their level of risk across the development lifecycle, reduce cost and improve software quality.

In September 2008, Experimentus became the first UK organisation to be awarded accreditation by the TMMi® Foundation for its in-house developed TMMi assessment method.

In December 2008, the Experimentus team became the first consultants worldwide to become accredited TMMi Assessors.

Workshop Overview

The aim of this workshop is to provide background and to help the attendees understand how to deliver qualitative and quantitative process improvements using the TMMi model. It explains in detail what TMMi is, and why it is different from other models. To help understand the benefits of the TMMi model, the day will also include a quick assessment which will provide an indicative view of where within the 5 levels of TMMi, each attendees company, project or team is currently positioned.
Throughout the day we will relate our experiences from the delivery of numerous Experimentus TMMi assessments and provide practical remedies and strategies to make an effective difference.
Finally there is a review of a recent TMMi survey undertaken by Experimentus to obtain a view of the trends in the software testing industry today.

To register please download the form at: http://www.testingexperience.com/TMMiWorkshop.pdf

£ 500

(plus VAT - £ 575)

testing experience

Knowledge Transfer

Experimentus
excellence in software quality management

Security Testing and Risk Management

by Eric Jarvi

A recent editorial in an IT trade magazine called on executives to push risk management into every level of their business. It prompted me to think about how testing can help with risk management at the software level of a business. One traditional view of risk management involves identifying hazards, assessing risk, and making decisions to act based on that information. It seems obvious that security testing is a big part of making that risk management process work for software. If it doesn't happen, there are blind spots. Certain risks go unidentified – meaning they can't be assessed, evaluated, or factored into any risk management decisions.

More often than not, software security risk is combined with financial, regulatory, legal, and other risk. The shared need for risk management creates an opportunity to explore how security testing fits into the picture, and to see if there are ideas we can bring over from other disciplines to do it better.

Software security testing is an emerging discipline with a colorful history. However, most historians would probably agree that the underlying problem of "risk identification" is of more ancient origin. Businesses have always needed some form of risk management function. In this sense, there may be something we can gain by learning from the ways engineers have faced risk management challenges in the past.

Some of NASA's historical challenges were dramatized in the contemporary film, *Apollo 13* . [1,2] Quotes from this movie seem almost identical to what could be expected of a software engineering team that realizes they're being exploited by a vulnerability they don't yet understand:

- "Let's work the problem people. Let's not make things worse by guessing." Gene Kranz (played by Ed Harris)
- "All right, we're not doing this, gentlemen. We are **not** doing this. We're not going to go bouncing off the walls for ten minutes, 'cause we're just going to end up back here with the same problems! Try to figure out how to stay alive!" Astronaut Jim Lovell (played by Tom Hanks)

The case can be made that systematic approaches and disciplined problem solving are sometimes our best option in chaotic, stressful, and unpredictable environments. Complex security issues often demand that we thoroughly "work the problem."

Working the Problem

Great thought has already gone into the space of "working" the software security problem. Much of it has been oriented towards development rather than test, but it is still important to understand. If you are new to the idea of threat modeling or haven't yet explored the concepts of the Security Development Lifecycle (SDL), you would be better off to stop reading this article and pick up the latest version of the Microsoft SDL and learn as much as you can about Threat Modeling. [3,4] These are great fundamentals, designed for almost universal applicability in software teams, and are often considered to be prerequisites for any type of security testing effort.

Much of that guidance was developed at a high level and puts a priority on helping developers and designers avoid introducing security holes in the first place. Defect prevention is a commendable priority, but it shouldn't lull testers into a false belief that the goal was actually accomplished, or worse yet, that no testing needs to be performed! If anything, defect prevention efforts make the challenge of defect detection all the more rewarding.

Basic Risk Management

Risk management, at a very basic level, can be described as a combination of risk identification, risk assessment, and risk evaluation. If you're attending a bug triage meeting, or debating whether or not a bug is worth fixing, you're actually doing an informal type of risk management on your software.

Sometimes there are more formal "risk management" processes in place. It is worth understanding what these processes are. In some cases, test teams can make a case for linking security testing work to the risk management function of the business and find more funding and support for it. At the very least, thinking about security testing from the risk management perspective of the organization may shine a light on vulnerabilities that weren't otherwise visible.

Risk Identification

Before we can manage or evaluate risks, we need to be able to identify them. Before we can talk about how bad a security bug is, we have to find it. Many standard techniques for identifying risk involve a variety of top-down approaches often adopted by large organizations:

- Checklists (For example, SDL 3.2 , .NET Framework Security) [5,6]

- Brainstorming sessions
- Audits (for example, WPF Critical Code Management code audits) [7]
- Subjective prioritization
- Risk Mapping (security visualizations) [8]

These types of approaches are often mandated and applied across the breadth of an organization, and we all generally become more familiar with them. Some approaches require going into depth. These often require technical expertise and low level investigation. HazOps, FMEA, and CCFA are three risk identification techniques that seem particularly interesting but lesser known. They demonstrate ways that engineers can leverage their technical depth to identify risk. Software testers may be able to leverage their technical depth in similar ways.

Risk Assessment

Once risk is identified, a risk assessment function evaluates both the frequency and consequence of a hazard. It can be a struggle to balance qualitative and quantitative metrics at this point. Multiple risk assessment functions can be used and factored in as part of the subsequent risk evaluation.

What's the frequency?

Higher frequency leads to higher risk. There have been times when a bug may seem insignificant in comparison with other bugs that need to be fixed. However, the frequency of that bug may be the real problem. I've heard this referred to as "death by 1,000 paper cuts." Attack surface reduction and "off by default" may be examples of ways to lower risk by lowering the frequency of the hazard.

If a bug triage process does not take frequency into account, a bug with a low consequence but high frequency might slip through. Teams may have to compensate with time and resources they don't have in the budget, or by shipping patches they weren't planning on having to release. Taking into account the frequency aspect is critically important, whether that means giving special consideration to frequency when defining a bug bar, adding a special field in the bug database so higher frequency bugs rise to the top of the list, or by otherwise adjusting the methodology or processes used by the team.

What are the consequences?

Naturally, a higher magnitude consequence will also increase risk. Being able to demonstrate the consequences of a particular bug is the second main factor in any risk management decision formula. Being able to communicate consequences factually, persuasively, and in detail is a hallmark trait of software testers who don't only log bugs, but actually get high numbers of them fixed.

The "Damage Potential" category of the DREAD [9] risk model offers an example of an attempt to include the notion of consequence in the quantification of risk. Knowing which areas have the highest consequence can help you identify areas to focus on with security testing.

Risk Evaluation

Risk evaluation follows risk identification and risk assessment. Risk evaluation provides information that can be used to make risk management decisions. Risk evaluations take into account the risks that were identified and the qualitative and quantitative assessments of their frequency and consequence.

Identification => Frequency Assessment + Consequence Assessment => Risk Evaluation

Software testers are familiar with variations on this concept. For example, a tester will typically begin at a state of not knowing where to start, employ techniques to identify bugs, and make an evaluation in the form of a bug report. This and other information is brought to bug triage, and decisions about what to do about the bugs are made. Security vulnerabilities catch a lot of attention when they go through this process, and the better information the tester can provide, the better the risk evaluation ends up being. A useful reference for tracking this type of information in bug reports is found at the end of the Microsoft SDL documentation in Appendix B: Security Definitions for Vulnerability Work Item Tracking. [10]

Finding Vulnerabilities

The point of this article is to help testers think about the way they go about finding security vulnerabilities. Risk identification in traditional risk management is similar in concept to discovering vulnerabilities in software testing. With this in mind, we can revisit a few techniques that are used for risk identification by other disciplines:

HazOps

As the story goes, the chemical process industry started to more widely accept what is known as the HazOps technique after a chemical plant explosion in 1974 killed 28 people. HazOps stands for Hazards and Operability Studies. [11,12] Both words have special meaning. In the chemical process industry, "Hazards" are defined as the operations of a system that can result in truly catastrophic consequences. In the software industry, "hazards" might be the high severity vulnerabilities found on the SANS Top-20, BugTraq, or US-CERT bulletins. [13,14,15] The term "operabilities," however, represents a less catastrophic form of risk. They might lead to shutdowns, regulatory violations, or other costly side effects. In software, these might resemble privacy, accessibility, reliability, performance, regulatory, or geopolitical issues.

Both hazards and operabilities are identified by bringing groups of technical experts together and looking for deviations between the implementation and the specification. It's a bottom-up process.

This is something that can be done in security testing as well. You don't get something for nothing, but it works. Read the spec in detail, and then look for inconsistencies with the implementation. Or, take snapshots of system state before and after exercising certain functionality and then "diff" to drill in further to find the areas most worth testing. Data flow diagrams of software can map the system, similar to plant schematics used in HazOps. Threat modeling often flushes out problems that are not catastrophic, but even those quirks pay off, in the sense that they help us build better mental models of failure modes that in turn can help us spot the vulnerabilities that exist.

If you need to sell management on an exercise like this, it's important to note that involving functional testers in security testing exercises like this can have virtuous side effects. In the typical HazOp study, more operabilities than hazards will emerge. In an equivalent software "HazOp" study, each stakeholder would similarly leave the exercise with a deeper understanding of the system and its limitations. But that knowledge is power and carries over to other day-to-day work. It can help educate teams and help stakeholders drive quality upstream in future discussions and design decisions, even though they may be unrelated to security.

FMEA

Failure mode and effects analysis (FMEA) [16] is an approach that was developed to identify the risk of defects and failure in the aerospace industry. This is another bottom-up risk assessment tool that leverages the deep understanding engineers have of failure modes and their ability to change the development process to avoid future failures. FMEA has also been used in military, automotive, and manufacturing environments to reduce the risk of future failures, and is finding more and more use in software development.

James Whittaker's How To Break Software [17] series offers an introduction to the types of faults and failure modes that software testers are familiar with. "The Practical Guide to Defect Prevention," [18] published by MSPress, actually includes an entire chapter on FMEA and its relationship to Fault Tree Analysis (FTA) and Failure Modeling along with an example of how it might be applied. Fault Trees, often referred to as "attack trees" during penetration tests, have been covered in detail by many in the security community, including Bruce Schneier. [19] Much of the common sense thinking behind FMEA carries over to traditional threat modeling and the concept should be familiar to anyone using the SDL.

Being able to think about software in terms of failure models and fault trees is a skill that can give software testers greater confidence when applying the SDL and can be a way to communicate risk more effectively to others in the organization. It may be called by other names than FMEA, but mastering this underlying approach to risk identification may help you flush out vulnerabilities that would have otherwise gone undetected.

CCFA

Common Cause Failure Analysis is interesting in that it takes the results of a top-down fault tree analysis as a prerequisite. The tester then walks back up the tree searching for coupling factors that could cause component failures to be interdependent on each other. "Common Cause Failure" and "Common Failure Mode" are other terms that are associated with this type of risk identification.

Variations on the CCFA technique may be particularly useful in environments that consist of integrated components that were not designed or even intended to ever work together. In some cases these components may have vulnerabilities that are only exploitable when combined with other components or used in unanticipated ways. In other cases, the behavior of one component may set the stage for the exploitability of a second component. For example, one application might assume that all files on the user's desktop are trusted binaries that can be executed, and another application might assume that the user's desktop is a safe place to silently drop files from the Internet.

CCFA may give security testers a way to tease more results out of fault tree work that was already done. Try working backwards up the tree to identify coupling factors with other components that might be unanticipated and could cause severe failure and vulnerability scenarios.

Conclusion

At its core, finding vulnerabilities is a risk identification problem and relied on by higher level risk management processes that govern our use of software. It can be possible for testers to make solid contributions to the risk management function of their organization by building a toolbox of proven approaches and defensively finding security vulnerabilities before they can be exploited.

If, as a tester, you're not having success finding vulnerabilities, perhaps some of the risk identification tools being used by other engineering disciplines will help you think about the problem in a new way.

[1] Apollo 13 Film, http://en.wikipedia.org/wiki/Apollo_13_(film)

[2] NASA SP-4223 "Before This Decade is Out..." http://history.nasa.gov/SP-4223/ch6.htm

[3] Microsoft SDL, http://go.microsoft.com/?linkid=8685076

[4] Microsoft Threat Modeling, http://msdn.microsoft.com/en-us/security/aa570413.aspx

[5] SDL 3.2, http://msdn.microsoft.com/en-us/library/aa302335.aspx

[6] .NET Framework Security, http://msdn.microsoft.com/en-us/library/cc307748.aspx

[7] Code Audit Example, http://msdn.microsoft.com/en-us/library/aa969774.aspx

[8] Security Visualizations, http://secviz.org/

[9] DREAD, David LeBlanc http://blogs.msdn.com/david_leblanc/archive/2007/08/13/dreadful.aspx

[10] Appendix B: Security Definitions for Vulnerability Work Item Tracking, http://msdn.microsoft.com/en-us/library/cc307392.aspx

[11] Lihou, Mike. Hazard and Operability Studies, http://www.lihoutech.com/hazop.htm

[12] Hazard and Operability (HazOp) Studies, http://pie.che.ufl.edu/guides/hazop/

[13] SANS Top-20 Security Risks, http://www.sans.org/top20/

[14] Security Focus Archive, http://www.securityfocus.com/archive/1

[15] US-CERT, http://www.us-cert.gov/cas/bulletins/

[16] Failure Modes and Effects Analysis (FMEA), American Society for Quality, http://www.asq.org/learn-about-quality/process-analysis-tools/overview/fmea.html

[17] How to Break Software, http://portal.acm.org/citation.cfm?id=515499

[18] The Practical Guide to Defect Prevention, http://www.microsoft.com/MSPress/books/9198.aspx

[19] Attack Trees, http://www.schneier.com/paper-attacktrees-ddj-ft.html

Biography

Eric Jarvi is a Security SDET on the Microsoft Office Trustworthy Computing Security Test Team in Redmond, Washington.

testing experience

Knowledge Transfer

— TMMi Foundation —

limited places

One Day Tutorial with **Erik van Veenendaal**

April, 1st 2009 in Frankfurt
June, 3rd 2009 in Stockholm

This workshop brings the TMMi model to Europe and is your chance to learn about the latest initiative in test process improvement.

The Test Maturity Model Integration is rapidly growing in use across Europe and the USA. It has been developed to compliment the existing CMMI framework. Its growing popularity is based upon it being the only independent test process measurement method, and the simple presentation of maturity levels that it provides.

In Europe the independent TMMi Foundation initiative has been established with the sole intent of elaborating the TMMi standard and developing a standard TMMi assessment and certification method, with the aim of enabling the standards consistent deployment and the collection of industry metrics.
Erik van Veenendaal has much practical experience in implementing the model and helping organisations improve the way they test, and the benefits this can generate. He is the editor of the TMMi Framework model and vice-chair of the TMMi Foundation. The workshop will present these experiences and benefits with the aim of providing the attendees with the information required to justify a test process improvement project.

As well as learning more about the TMMi during the workshop each attendee will be provided with the information and practical materials needed to do an evaluation of their own companies test maturity level.

Key learning points

- Understanding the objectives and (intermediate) results achieved in the TMMi Foundation
- Understanding of the TMMi model and its practical implementation
- Practical application of the TMMi assessment techniques

To register send an e-mail to kt@testingexperience.com

750,- €
(plus VAT)

Testing & Finance 2009

Early Bird ends by May 1st, 2009

The Conference for Testing & Finance Professionals

June 22nd - 23rd, 2009 in Bad Homburg (near Frankfurt am Main, Germany)

The international well-known speakers are amongst others:

Prof. Dr. Schulte-Mattler, Dr. Mike Bartley, Dorothy Graham, Rex Black, Erik van Veenendaal, Graham Bath, Vipul Kocher, Manu Cohen-Yashar, Hans Schäfer, Alon Linetzki, Yaron Tsubery

www.testingfinance.com

Exhibitors

Saxonia Systems
A vision of technology.

Compuware

hp

SELA GROUP

PureTesting
Testing Thought Leadership

CaseMaker

ib-bank-systems

iSQI
International Software Quality Institute

Supporting Organisations

VOB SERVICE

dpunkt.verlag

Díaz Hilterscheid

te testing experience

Testing & Finance 2009

The Conference for Testing & Finance Professionals

June 22nd - 23rd, 2009 in Bad Homburg
(near Frankfurt am Main, Germany)

Please fax this form to +49 (0)30 74 76 28 99 or send an e-mail to info@testingfinance.com.

Participant

Company: _____
First Name: _____
Last Name: _____
Street: _____
Post Code: _____
City, State: _____
Country: _____
Phone/Fax: _____
E-mail: _____

Early Bird ends by May 1st, 2009

Billing Address (if differs from the one above)

Company: _____
First Name: _____
Last Name: _____
Street: _____
Post Code: _____
City, State: _____
Country: _____
Phone/Fax: _____
E-mail: _____
Remarks: _____

☐ 1 Day Early Bird **375,- €** (plus VAT)
450,- € (plus VAT)

☐ 2 Days Early Bird **700,- €** (plus VAT)
850,- € (plus VAT)

☐ Yes, I like to join the social event on June 22nd, 2009.

Included in the package: The participation on the exhibition, at the social event and the catering in course of the event.

Notice of Cancellation

No fee is charged for cancellation up to 60 days prior to the beginning of the event. Up to 30 days prior to the event a payment of 50% of the course fee becomes due and up to 15 days a payment of 100% of the course fee becomes due. An alternative participant can be designated at any time and at no extra cost.

Settlement Date

Payment becomes due no later than the beginning of the event.

Liability

Except in the event of premeditation or gross negligence, the course holders and Díaz & Hilterscheid GmbH reject any liability either for themselves or for those they employ. This also particularly includes any damage which may occur as a result of computer viruses.

Applicable Law and Place of Jurisdiction

Berlin is considered to be the place of jurisdiction for exercising German law in all disputes arising from enrolling for or participating in events by Díaz & Hilterscheid GmbH.

Date Signature, Company Stamp

»Let's Talk About Testing«

An imaginary talk between Herbert D. Benington and Andreas Spillner

by Andreas Spillner

Andreas: Hi Herbert, how are you? What's going on in your job?

Herbert: Hi Andreas, nice to meet you. I'm fine, thanks!

I'm working at MIT's Lincoln Laboratory and we produce programs for the SAGE (*Semi-Automatic Ground Environment*) air defence system.

A: That sounds very interesting. Tell me about your lessons learned.

H: Yes, I'm up for it! Let's start with the most unpopular part of the job.

There is the problem of *documentation*. In the early days of programming, you could call up the programmer if the machine stopped. These days there is a long way to go before we can take an integrated program of 100,000 instructions and make it "public property" for the user, the coder, the tester, the evaluator, and the on-site maintenance programmer. The only answer seems to be the documentation of the system on every level from sales brochures for management to instruction listings for maintenance engineers.

A: I don't know anyone who likes to write documentation, and I haven't seen a lot of good descriptions of using tools or a web-interface. Another problem is that often tools don't work the way which is described in documentation. Users have to decide whether the documentation or the program is wrong.

H: Probably many of the decentralized systems currently in operation still contain many minor errors, which are being compensated for daily by users who have become accustomed to these minor idiosyncrasies.

A: These programs are not tested or not tested enough? What are you doing at MIT's Lincoln Laboratory to test the software and find all these bugs?

H: After coding, each component subprogram is *parameter* tested on the machine by itself. This testing phase uses an environment that simulates pertinent portions of the system program. Each test performed during this phase is documented in a set of test *specifications* that detail the environment used and the outputs obtained.

A: You are testing the units of the system and use different values of the parameters passed over the unit interface, that's why you call it parameter testing. What's the basis for specifying these tests?

H: Parameter testing is guided by the coding specifications instead of by the coded program; in other words, a programmer must prove that he satisfied his specifications, not that his program will perform as coded.

A: You are no friend of white-box testing, I agree. When do you start writing test cases?

H: Actually, test specifications for one subprogram can be prepared in parallel with the coding.

A: Testing and coding in parallel, that's just like in the "Test-First" approach. What are next steps after parameter testing?

H: As parameter testing of component subprograms is completed, the system program is gradually *assembled and tested*, using first simulated inputs and then live data. For each test performed during this period, *assembly test specifications* are prepared that indicate test inputs and recorded outputs. Assembly testing indicates that a system program satisfies the operational and program specifications.

A: Stepwise integration and test to find bugs which cannot be found by unit testing. Well, we use the same strategy, followed by system testing.

H: When the completed program has been assembled, it is tested in its operational environment during *shakedown*. At the completion of this phase, the program is ready for operation and evaluation.

A: We call these levels component test, integration test and system test. The evaluation is the acceptance test performed by the customer. Do you use a tool to localize the bugs?

H: In order to debug programs, a "checker" facility is used. This is a service program of 10,000 instructions that allows the program to be tested - the checkee - to be operated either interpretively or noninterpretively under control of a pseudoprogram of executive instructions. When the checkee is operated in the interpretive mode, the checker automatically detects loops, arithmetic alarms, illegal in-out sequences, and illegal instructions. It stores a history of program operation including branches, change-registers, and in-out transfers.

A: It seems this is not just a debugger, but also a static analysis tool.

H: Yes, it is. In the interpretive mode, the checkee provides a wide variety of outputs including instruction-by-instruction printouts, dynamic change-register printouts, and alarm printouts.

A: I see, Checkee provides you a lot of helpful information.

H: Yes, it helps a lot. Using the executive instructions, a programmer can set machine registers or memory registers to test values; he can start and stop the checkee at selected loca-

tions; he can request different outputs for different regions of the program; he can request alarm outputs if the checkee transfers control outside a fixed region or if a loop of more than n cycles is performed; he can indicate the use of different executive subprograms depending on the results of checkee operation; he can indicate which portions of his program are to be performed noninterpretively.

A: I understand, a good debugger with many options to control and monitor the program is very important support! But first you have to test extensively, to find all these stupid bugs.

H: It is debatable whether a program of 100,000 instructions can ever be thoroughly tested - that is, whether the program can be shown to satisfy its specifications under all operating conditions. For this reason, one must accept the fact that testing will be sampling only.

A: Are your saying to test only a little bit, only the main functions of the system?

H: No, not at all! Many sad experiences have shown that the program testing effort is seldom adequate. When the program is delivered for operation, its performance must be highly reliable because the control system is a critical part of a much larger environment of men and machines. One error per 100,000 operations of the entire program can easily be intolerable.

A: Yes, most of the programs are part of embedded systems, which must work reliable. We really have a responsible job. Do you have any principles for testing?

H: Yes, we have, let me tell you. As a result of facing this problem for some time at the Lincoln Laboratory, the following principles have evolved to govern our testing.

First, parameter testing (i.e., testing of individual component subprograms in a simulated environment) cannot be too thorough. This phase must discover all errors internal to the program and its individual coding specifications. Even if parameter testing were perfect (which it never is!), many errors in system design would remain to be discovered during subsequent assembly testing.

A: In our opinion performing only unit tests is inadequate, no matter how good and how many unit tests are executed. Please, tell me more about your testing principles.

H: Second, initial assembly testing should be performed using completely simulated inputs. There are several reasons. First, only in this way can all test inputs be carefully controlled and all tests be reproducible. Second, when errors are discovered with a new program using live inputs, there will always be a question whether the program or the machine is at fault. Integration of the system program with terminal equipment should not be attempted until the assembled program has been well tested.

A: Thanks for this very good explanation of why a step is needed between unit test and system test. And this step is a very important one.

H: Let me continue. A third principle is that the testing facility used during the assembly test phase must contain extensive, flexible facilities for recording both system outputs and intermediate outputs (i.e., subprogram intercommunications). Without this facility, rapid and reliable diagnosis of system errors is impossible. After a test has been conducted and errors found, it should be possible to correct the error before the program is put on the machine again.

A: I wish that many more programmers and testers follow your testing principles and that managers provide the necessary resources. Are there any requirements how to build the program, something like "design for testability"?

H: The need for comprehensive simulated inputs and recorded outputs can be satisfied only if the basic design of the system program includes an instrumentation facility. In the same way that marginal-checking equipment has become an integral part of some large computers, test instrumentation should be considered a permanent facility in a large program.

A: What about regression testing? Are you prepared for maintenance testing?

H: Sure we are. One final principle should govern system-program testing: *All successful parameter and assembly tests must be reproducible throughout the life of the system program*. These tests must be documented in test specifications that detail the reasons for the tests, required inputs, operating procedures, and expected outputs.

A: "Conservation of the testware" is the term for these activities in my projects.

H: The original reason for this requirement stemmed from the problem of revising the program once it was operational. The slightest modification to a program can be successful under limited testing conditions and yet still cause critical errors for other operations. Since it is desirable to retest the program thoroughly after each modification, a large battery of test inputs must be available. We have discovered two other incidental advantages of detailed test documentation. First, a programmer's tests tend to be more organized and more exhaustive if he must document them. Second, if machine-versus-program reliability is ever questioned, retesting is possible. If a known program and a known test fail, the machine is at fault.

A: The amount of regression testing is always a problem. How do you deal with new or changed user requirements?

H: Consider the problem of revising the system once the program is operational in the field. A minor change in the operational specifications is proposed. First, the cost and effects of this change must be evaluated in terms of the program, the operators, and, often, the machine. In order to make the change, several hundred revisions may be required in the specifications. If the change is approved, these documents must be changed, operating manuals revised, and the program modified and thoroughly tested. The wave of changes must be coordinated smoothly.

A: I like your last sentence. I hope we always have customers, who are "producing" small waves and not tsunamis. Can you sum up your experiences in some short statements?

H: As a result of progress in this area (and a growing number of experienced programmers), we find that large programs can now be produced; unfortunately, they are difficult to test and document. If the newest very-high-speed, large-memory computers are to be fully utilized, we must develop automatic programming procedures so that they allow cheap production of highly reliable, easily revised, well-documented system programs.

A: You see the future of programming in generating code out of some kind of formal specification, like UML. Well, since a few years we have these kinds of tools. Let me put my last question: Which kind of computer do you use?

H: We use the Whirlwind I, FSQ-7.

A: I never heard of this machine or computer. Is it a new one, or is it secret, because it is only used by military?

H: No, the FSQ-7 was the largest machine we felt able to build in the early 1950s.

A: In the early 1950s? Come on, that's got to be a joke!

H: No, it's not a joke. All my answers are quotation from my old paper titled "Production of Large Computer Programs". The paper was presented in Washington, D.C., in June 1956 at a symposium on advanced programming methods for digital computers, sponsored by the Navy Mathematical Computing Advisory Panel and the Office of Naval Research. You can find a reprint of this paper in IEEE Annals of the History of Computing, Volume 5, Issue 4, October 1983, pages 350-361, ISSN: 1058-6180

In the Internet: http://portal.acm.org/ and search for the title of my presentation.

A: I thought all your great ideas about testing came from the last ten or twenty years, but not older than 50 years! Perhaps it's a good idea to read some famous old papers …

Biography

Andreas Spillner is professor at the Hochschule Bremen (University of Applied Sciences). He is a founder member of the German Testing Board e. V. and was founder and chair of the German Special Interest Group in Software Testing (SIGIST, »Test, Analyse und Verifikation von Software«, 1990-2003). He is a fellow of the German Informatics Society (GI - Gesellschaft für Informatik, 2007)

He is author of German and English books and publications, and speaker at many national and international conferences. For detailed information take a look at his home page (www.informatik.hs-bremen.de/spillner).

Improve your professional life!

Check our website on April 14th

www.testingexperience.com

te testing experience

Interview Doron Reuveni

Doron Reuveni, founder and CEO of uTest and
José Díaz, Editor

Could you explain to the readers what uTest is?

uTest is the world's largest marketplace for software testing services. We've built a community of 13,000+ professional testers from 149 countries around the world. Software companies turn to uTest to test their web, desktop, mobile and gaming applications. uTest enables these companies to compliment their in-house QA team with on-demand testers that can cover all locations, languages and technical platforms.

Why should I - as tester - work for uTest?

uTest enables testers to stay in touch with the latest web, mobile and gaming apps, as well as to earn extra money. Plus, uTest fits your schedule – so you can participate in a uTest project when it matches your ability and willingness to offer professional testing services. That said, the more you participate, the better reputation you'll build, and the more work you'll win in the future.

An additional benefit of joining uTest is that it puts you in touch with a community of your professional QA peers. We already enable a free exchange of testing ideas, best practices and news and in 2009, we're investing even more in our community – training and certification programs, tester recognition programs, QA-focused webinars, content and contests.

What is the added value for a company to work with uTest?

uTest enables companies to launch higher quality applications; to get to their products to market faster; and to get the testing coverage they need across platforms, locations and languages. And because we work so closely with our customers' in-house QA resources, uTest enables to do all this without adding any fixed cost or additional headcount.

Quality Assurance and Testing is more complex than just finding bugs. How does the process work? How does the company really get the most out of that?

Not always is a supposed bug really a bug uTest employs a pay-per-bug model, which means that our customer pay only for those bugs they approve as valid. uTest also offers professional project management services, which help customers to write effective test plans, manage their projects and review lists of reported bugs. This uTest project manager closely partners with the QA manager from our customer to ensure they get the testing coverage and results they require.

How do you assure that all the testers involved use the same techniques or tools?

Actually, we purposely don't require all testers to use the exact same techniques or tools. With a model like uTest, it's the diversity of approaches, techniques and tools which enable us to offer unbeatable testing coverage. This is one of the things that our customers love about uTest.

How do you control or measure the quality or skills of the testers? Does it help you or your customers to know that they are ISTQB, QAMP certified?

In an open marketplace like uTest, testers build reputations based on their past performance. So the uTest marketplace between companies and testers becomes a living, breathing, self-policing entity which promotes good behavior and ensures that quality work is recognized and rewarded.

As far as external certification, we are actually in the process of launching several tester certification programs in our marketplace right now. This will enable our testers to sharpen their skills, earn certifications and differentiate themselves within our community.

How does the information flow between the testers and between testers and companies?

Bugs and or completed test scripts are submitted by testers and logged in the secure uTest platform. When the customer logs in to their account, they see the bugs that have been reported within their project. We also enable online communication between customers and testers during a release cycle, so if a customer wants to follow up with a tester about a specific bug, they can communicate live through uTest.

How do you assure that the regression tests can be done by other testers with other skills?

Every release cycle executed with uTest has a specific focus, as defined by the customer. The focus can be on particular areas, running test scripts or general testing coverage. Conversely, many uTest customers run full regression for their applications through our community.

When doing full regression testing, customers can request that all previously submitted bugs be verified and re-checked by a certain set of testers, or even by the exact same testers that originally submitted each bug.

How do you create the documentation? Do you have a standard?

uTest uses a standardized methodology for performing a testing project with uTest. We've based this methodology on best practices derived from years of experience testing desktop, web and mobile apps. Every

testing project that is run through the uTest marketplace is managed using a standard template, and every bug is submitted through a standardized template & interface.

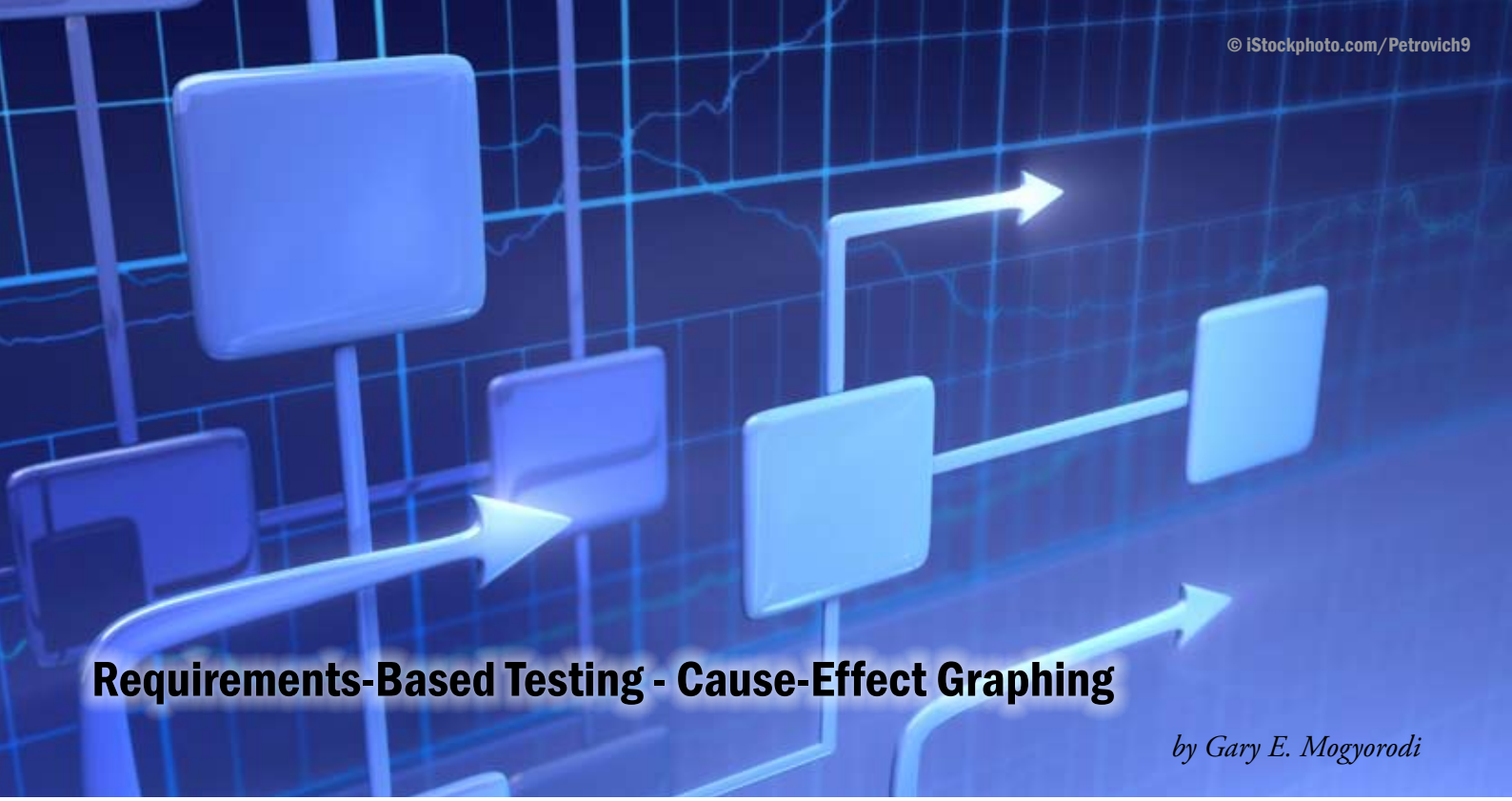
Further, uTest customers have the capability to define the structure and the coverage report of the testing they would like our community to perform. uTest also provides a dedicated project manager to our customers to help facilitate their test cycles. This helps ensure that their testing needs are met, and they get the most value of testing with our community.

Do you think that today – in the crazy crisis world – there is a chance for uTest to grow better than in other times?

A marketplace is the most efficient method of delivering this type of real-time, real-world testing service – that's why we're popular with our customers. Thus, we've seen that in good times, software companies are attracted to uTest because they can't find enough qualified QA professionals to adequately test their apps. And in tough economic times, they flock to us because they're trying to contain costs and do more with less.

Is uTest an option for big companies too or only for small companies with small projects?

Companies of all shapes and sizes are now using uTest – from five-person startups to Fortune 500 enterprises. We've found that many of our customers employ agile development methodologies, which implies start-ups, but many enterprise dev teams are shifting to more fluid, agile development structures. And even those firms that use more traditional waterfall development methodologies find that they need testing coverage that spans across locations, languages and platforms.



Requirements-Based Testing - Cause-Effect Graphing

by Gary E. Mogyorodi

This article provides an overview of the Requirements-Based Testing (RBT) process. RBT is a rigorous technique for improving the quality of requirements, while deriving the minimum number of test cases to cover 100% of those requirements. RBT is comprised of two techniques: Ambiguity Reviews and Cause-Effect Graphing.

An Ambiguity Review is used in the requirements phase of software development to identify ambiguities in functional1 requirements. The intent of an Ambiguity Review is to identify anything that is unclear, not concise or ambiguous in the requirements. The elimination of these ambiguities improves the quality of those requirements.

Cause-Effect Graphing is a test case design technique that is performed once requirements have been reviewed for ambiguity, followed by a review for content. Requirements are reviewed for content to insure that they are correct and complete. The Cause-Effect Graphing technique derives the minimum number of test cases to cover 100% of the functional requirements to improve the quality of test coverage.

This article addresses the second RBT technique - Cause-Effect Graphing. The first article described Ambiguity Reviews and included a sample ambiguity review of the requirements I will design test cases for in this article.

Test Case Design

The test case design challenge is to derive the necessary and sufficient set of test cases to cover 100% of the functionality for the system under test. Without 100% functional coverage, there is a certainty that functionality will go into production untested.

There are many test case design techniques, but few insure that the test cases will provide 100% functional coverage. The Cause-Effect Graphing technique begins with the set of requirements, and determines the minimum number of test cases to completely cover the requirements.

Test Case Review

Once the test cases have been derived by the test case designer, the RBT process includes the review of these test cases by the following stakeholders:

- The author of the requirements verifies that he/she agrees with the test case designer's translation of requirements to test cases.
- The domain experts review the test cases in order to determine the answer to the following question: "Are we building the right system"?
- The developers review the test cases to clarify their understanding of the requirements. The developers learn what they will be tested on, and can therefore develop the software to succeed.

By performing these reviews, everyone on the project team can obtain the same understanding of what will be built.

Traditional Test Case Design Techniques

There are a number of test case design techniques, such as Equivalence Class Partitioning and Boundary Value Analysis. These techniques rely on the test case designer to manually work out the proper combinations of test cases. Often, the test case designer does not use a formal test case design technique and relies on his/her "gut feel" to assess whether test coverage is sufficient.

While these techniques do generate combinations of test cases, they often fall short on providing full functional coverage. Too often the normal flow or "go path" functionality has overlapping, redundant test cases, while exceptions and error conditions go untested.

Cause-Effect Graphing

The Cause-Effect Graphing technique was invented by Bill Elmendorf of IBM in 1973. Instead of the test case designer trying to manually determine the right set of test cases, he/she models the problem using a cause-effect graph, and the software that supports the technique, BenderRBT, calculates the right set of test cases to cover 100% of the functionality. The cause-effect graphing technique uses the same algorithms that are used in hardware logic circuit testing. Test case design in hardware insures virtually defect free hardware.

Cause-Effect Graphing also has the ability to detect defects that cancel each other out, and the ability to detect defects hidden by other things going right. These are advanced topics that won't be discussed in this article.

The starting point for the Cause-Effect Graph is the requirements document. The requirements describe "what" the system is intended to do. The requirements can describe real time systems, events, data driven systems, state transition diagrams, object oriented systems, graphical user interface standards, etc. Any type of logic can be modeled using a Cause-Effect diagram. Each cause (or input) in the requirements is expressed in the cause-effect graph as a condition, which is either true or false. Each effect (or output) is expressed as a condition, which is either true or false.

Don’t forget it!

Improve your professional life!

Check our website on April 14th

www.testingexperience.com

te testing experience

A Simple Cause-Effect Graphing Example

I have a requirement that says: "If A OR B, then C."

The following rules hold for this requirement:

- If A is true and B is true, then C is true.
- If A is true and B is false, then C is true.
- If A is false and B is true, then C is true.
- If A is false and B is false, then C is false.

The cause-effect graph that represents this requirement is provided in Figure 1. The cause-effect graph shows the relationship between the causes and effects.

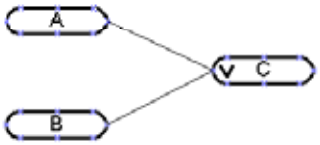


Figure 1 – A Cause-Effect Graph

In Figure 1, A, B and C are called nodes. Nodes A and B are the causes, while Node C is an effect. Each node can have a true or false condition. The lines, called vectors, connect the cause nodes A and B to the effect node C.

All requirements are translated into nodes and relationships on the cause-effect graph. There are only four possible relationships among nodes, and they are indicated by the following symbols:

Where A always leads to C, a straight line ----------.

Where A or B lead to C, a V at the intersection means "or".

Where A and B lead to C, an inverted V at the intersection means "and".

A tilde ~ means "not" as in "If not A, then C".

The Decision Table

The cause-effect graph is then converted into a decision table or "truth table" representing the logical relationships between the causes and effects. Each column of the decision table is a test case. Each test case corresponds to a unique possible combination of inputs that are either in a true state, a false state, or a masked state (the masked state will be described in Figure 4 below).

Since there are 2 inputs to this example, there are 2 ** 2 = 4 combinations of inputs from which test cases can be selected.

	Test#1	Test#2	Test#3
Causes:			
A	T	F	F
B	F	T	F
Effects:			
C	T	T	F

Figure 2 – The Decision Table

Figure 2 represents the decision table derived for the cause-effect graph in Figure 1.

Notice that there are only three test cases. However, these three test cases cover 100% of the functionality for this example. The fourth combination of inputs (A= true and B = true) does not add any additional functional coverage, and is a redundant test case. Each relation is tested with the right combinations of causes so that all defects are covered for that relation, resulting in 100% coverage.

Let's Create a Cause-Effect Graph

In the previous article on Requirements-Based Testing, we performed an Ambiguity Review and revised our original set of requirements to the following:

The Postal Regulation for South America (Version 1)

The following postal regulation determines postage surcharges. This regulation applies only to parcels, not other types of postal items, and only to parcels sent to South America. Other postal items are envelopes and post cards. There is no postage surcharge for envelopes and post cards. For parcels which people mail to South America between December 1 and December 24 of each year, the system will apply the surcharges in Table 1 in addition to the standard postage of $5.00 US:

Table 1 - Country & Weight Surcharge between December 1 and December 24

Argentina	All weights	$11 US
Brazil	Weight > 33 pounds	$21 US
Brazil	Weight = 33 pounds	$19 US
Brazil	Weight < 33 pounds	$17 US
Columbia, Ecuador, Peru, Bolivia, Chile, French Guiana, Guyana, Paraguay, Suriname, Uruguay, Venezuela, and Falkland Islands	All weights	$15 US

For parcels which people mail to South America outside of December 1 to December 24 of each year, the system will apply the surcharges in Table 2 in addition to the standard postage of $5.00 US:

Table 2 - Country & Weight Surcharge outside the dates December 1 to December 24

(In a real life problem, the specific details of Table 2 would be supplied here. For this example, they have been excluded.)

Figure 3 – The Postal Regulation Requirements Corrected for Ambiguity

For more technical information about Cause-Effect Graphing, refer to the following:

Richard Bender
Bender RBT Inc.
17 Cardinale Lane
Queensbury, NY 12804
518 743-8755
www.benderrbt.com

G. J. Myers,
The Art of Software Testing,
Wiley, 1979.

Cause-Effect Graphing Analysis and Validation of Requirements
Khenaidoo Nursimulu and Robert L. Probert
Bell-Northern Research and Telecommunications Software Engineering Research Group
Department of Computer Science
University of Ottawa, 150 Louis Pasteur/Priv., Ottawa
Ontario, Canada K1N 6NA
www.cs.ubc.ca/local/reading/proceedings/cascon95/pdf/nursimul.pdf

Cause-Effect Graphing
Theresa Hunt
http://www.westfallteam.com/Papers/Cause_and_Effect_Graphing.pdf

The test case designer translates the requirements into the cause-effect graph shown in Figure 4 (created using BenderRBT).

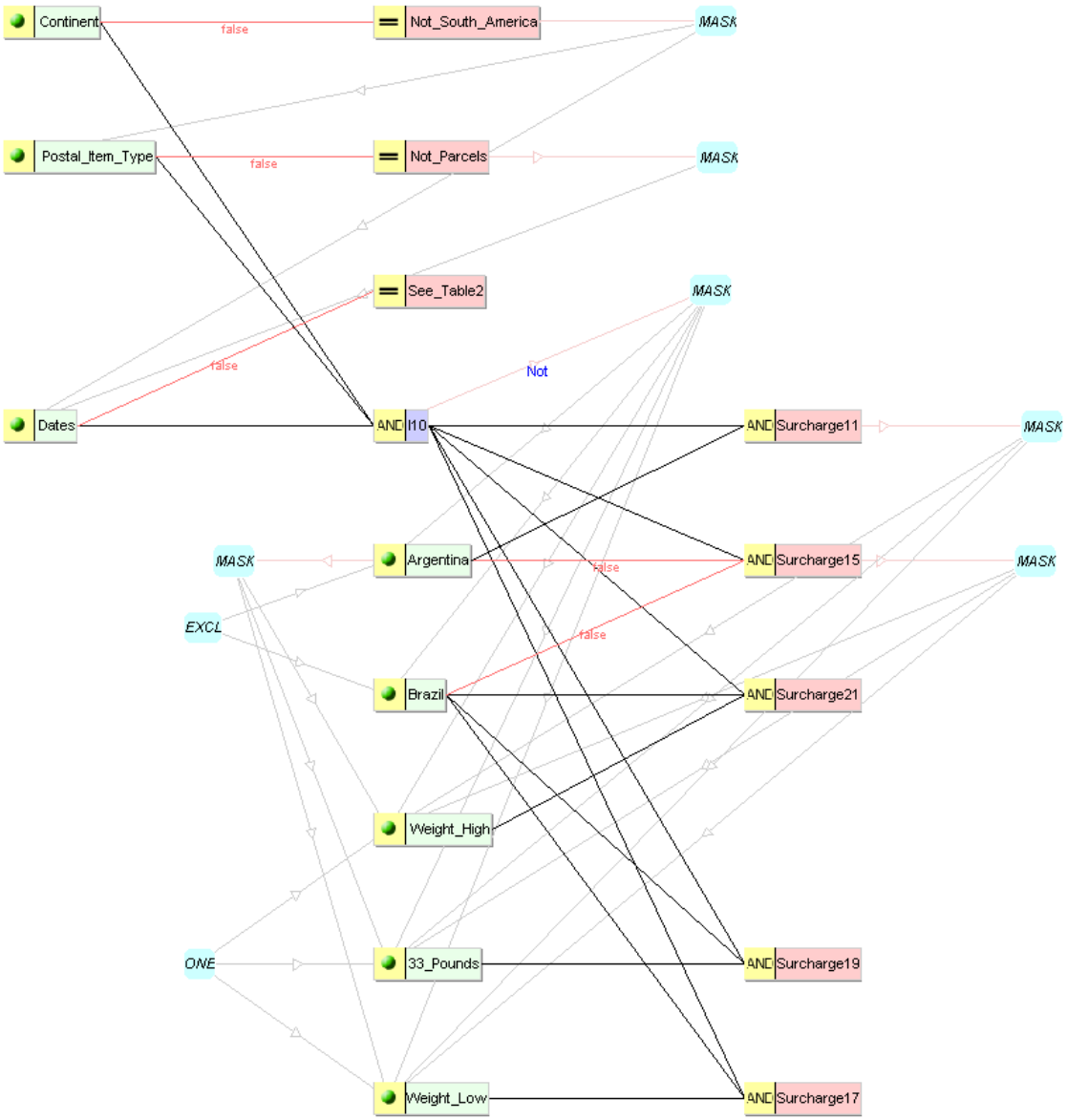


Figure 4 – Cause-Effect Graph for the Postal Regulation Requirements

		Test#1	Test#2	Test#3	Test#4	Test#5	Test#6	Test#7	Test#8
Causes:		T	F						
Continent		F	T	T	T	T	T	T	T
Postal_Item_Type		M	F	T	T	T	T	T	T
Dates		M	M	F	T	T	T	T	T
Argentina		M	M	M	T	F	F	F	F
Brazil		M	M	M	F	F	T	T	T
Weight_High		M	M	M	M	M	T	F	F
33_Pounds		M	M	M	M	M	F	T	F
Weight_Low		M	M	M	M	M	F	F	T
Effects:									
Not_South_America	{obs}	T	F	F	F	F	F	F	F
Not_Parcels	{obs}	m	T	F	F	F	F	F	F
See_Table2	{obs}	m	m	T	F	F	F	F	F
I10	<OBS>	F	F	F	T	T	T	T	T
$11	{obs}	F	F	F	T	F	F	F	F
$15	{obs}	F	F	F	F	T	F	F	F
$21	{obs}	F	F	F	F	F	T	F	F
$19	{obs}	F	F	F	F	F	F	T	F
$17	{obs}	F	F	F	F	F	F	F	T

Figure 5 – Decision Table for the Postal Regulation Requirements

The node I10 represents the state in which the item is sent to South America AND the postal item is a parcel AND the item is mailed between December 1 and December 24.

There are a number of mask statements in the cause-effect graph. The mask statements define constraints on the system under test. As an example, the mask (Argentina, Weight_High, 33_Pounds, Weight_Low) means that when Argentina is the country, then the weight divisions associated with Brazil are irrelevant inputs.

The Cause-Effect Graph is translated into the following decision table (output from BenderRBT):

Each column in Figure 5 represents a test case. The inputs are listed under the Causes, while the outputs are listed under the Effects. Each value in the matrix can be either T = true, F = false, or M = masked (an input that is masked is irrelevant to that test case). Each column represents a unique combination of inputs.

The following 8 test cases are derived for the Postal Regulation requirements using the Cause-Effect Graphing technique:

TEST#1 -- The Postal Regulation

Cause(s):

The item is NOT sent to South America

Effect(s):

These conditions are out of scope for the Postal Regulation problem.

TEST#2 -- The Postal Regulation

Cause(s):

The item is sent to South America

The postal item is NOT a parcel.

Effect(s):

These conditions are out of scope for the Postal Regulation problem.

TEST#3 -- The Postal Regulation

Cause(s):

The item is sent to South America

The postal item is a parcel.

The item is NOT mailed between December 1 and December 24.

Effect(s):

See Table 2 for postage surcharges.

TEST#4 -- The Postal Regulation

Cause(s):

The item is sent to South America

The postal item is a parcel.

The item is mailed between December 1 and December 24.

The country is Argentina

Effect(s):

The postage surcharge is $11

TEST#5 -- The Postal Regulation

Cause(s):

The item is sent to South America

The postal item is a parcel.

The item is mailed between December 1 and December 24.

The country is NOT Argentina

The country is NOT Brazil

Effect(s):

The postage surcharge is $15 US.

TEST#6 -- The Postal Regulation

Cause(s):

The item is sent to South America

The postal item is a parcel.

The item is mailed between December 1 and December 24.

The country is Brazil

The item weighs more than 33 pounds

Effect(s):

The postage surcharge is $21 US.

TEST#7 -- The Postal Regulation

Cause(s):

The item is sent to South America

The postal item is a parcel.

The item is mailed between December 1 and December 24.

The country is Brazil

The item weighs 33 pounds

Effect(s):

The postage surcharge is $19 US.

TEST#8 -- The Postal Regulation

Cause(s):

The item is sent to South America

The postal item is a parcel.

The item is mailed between December 1 and December 24.

The country is Brazil

The item weighs less than 33 pounds.

Effect(s):

The postage surcharge is $17 US.

In Summary

Figure 5 shows there are 8 inputs to the Postal Regulation requirements (i.e., Continent, Postal_Item_Type, Dates, Argentina, Brazil, Weight_High, 33_Pounds, Weight_Low). It is interesting to note that with 8 inputs, there are 2 ** 8 = 256 combinations of inputs from which test cases can be selected. The Cause-Effect Graphing technique derived only 8 test cases, but these 8 test cases cover 100% of the functionality described in the requirements. No other test cases are required to improve functional coverage. Additional test cases would only provide redundant coverage already supplied by some portion of each of the 8 original test cases.

Cause-Effect Graphing, in conjunction with a powerful analytical tool such as BenderRBT, provides a rigorous, consistent and cost effective approach to deriving test cases. Since the technique determines the minimum number of test cases, the test effort is minimized, and yet the test case designer can be confident that once these tests are successfully run, then testing is complete, and no untested functionality will move into production.

Notes: 1 Functional requirement - A functional requirement specifies what the system must be able to do, in terms that are meaningful to its users.

Biography

Gary Mogyorodi has over 30 years of experience in the computing industry. Currently, as Principal Consultant with Software Testing Services, he consults, trains and mentors in software testing, specializing in Requirements-Based Testing (RBT), i.e. Cause-Effect Graphing and Ambiguity Reviews. Some of his customers include CGI, FiLogix, H&R Block, Home Hardware, IBM, Manulife Financial, RBC, RBC Dexia, Siemens and TELUS.
Gary Mogyorodi began practicing RBT with Bender & Associates in 1998.
Mr. Mogyorodi is the President of the Canadian Software Testing Board. He is an approved ISTQB Instructor at the Foundation Level.

Thinking Outside Of The Box

or how to deal with the challenges of outsourcing

by Aleksandra Popara & Predrag Skokovic

Relying on somebody else, probably unknown and a long way off, to do business for your company is always tricky; especially if this is the first time you are seeking an outsourcing partner. Even though, an abundance of companies are turning to nearshoring/offshoring on a full-scale or outsourcing just part of their work. The main guiding ideas for companies who enter this area are certainly attractive prices & cost efficiency. On the other hand, some developed countries are facing a serious lack of educated engineers, developers, or testers. Regardless of the reasons, they all deal with the same issues: the so called 'culture soup'; different time zones, geographical distance, newcomers who are ready to adopt almost anything that smells of money…

So, why is the trend of outsourcing expanding every year? Moreover, it has become a standard business model, moving from simple 'brain' outsourcing to complete solutions development.

Concerning the fourteen years of experience our company has in outsourcing, we can say that it is an excellent way to go. If managed properly, of course. Unfortunately, there is no model for overcoming the obstacles mentioned above and no guaranty that the cooperation will be a winner, but there are a couple of important factors that must not be overlooked. Key points for making the nearshoring/offshoring partnership success work:

Communication as an indispensable instrument for surpassing cultural differences and misunderstandings. Sometimes even neighboring countries find themselves in this arena because a lot of the culture and business logic are not always clearly visible.

Does their 'Yes' actually means 'No'? Am I supposed to speak now or it is rude to interrupt the speaker?

These are the common questions. In order to have everybody on the same page, companies have to apply extra effort to this matter. First, it is important to make sure the teams practice regular on-site meetings, calls and have defined communication channels. Feedback on every matter is very precious. The second point, as important as the matter pointed out before, is investing in relationship building. After-work dinner or a town-tour are highly appreciated by a visiting company. It is called Relationship Management and gets you both to a position that ensures smoother project implementation. This way, the 'not invented here' attitude gets eliminated and both companies will manage to elevate their cooperation to a higher level – strategic partnership.

Expertise and quality of services provided will lead to ROI. It is highly appreciated if the outsourcing company has experience and requested domain knowledge. References and project lists can help in narrowing the partner selection and quality certificates are an additional pro for the cooperation. Another relevant index is the company's country competitive position in the global IT market. Regarding the costs, companies usually think that the spending is done once the contract is signed. Well guess what? Even when the model of cooperation (fixed price model, extended office model…) is chosen, they have to be prepared for some additional spending. Statistics say that an extra 1-10 percent of initial costs are spent on travel costs and in the first three to six months of project implementation; and 6-10 percent on managing the nearshore/offshore contract. Conclusion: Without proper expertise on both sides, one cannot count on getting the invested money out let alone make a profit!

Once the negotiations are over and it's time to start implementation, we put an accent to *Strong definition of project scope & timelines* in terms of clearly defining project requirements, testing requirements, protocols and delivery dates. Believe us, if anything can be misunderstood it will be. It is really important not to assume anything: always keep track of a detailed project plan and be informed on all the steps which are to be delivered, or carried out. In order to achieve this, participants have to make their project management as transparent as possible and actively interact with each other. When dealing with a company whose organizational skills are not that well developed, the partnering company must avoid being dragged down as well. Practicing discipline and taking the lead when necessary are definitely good choices for staying on the right track. Nearshoring/offshoring partners are in business together and must be evenly involved in project planning and realization.

Whole team engagement from the very beginning of project realization (on both sides!), meaning that managers, developers and testers have strong awareness of the need to share knowledge and responsibility. This requires that everybody in the teams understands project scope, requirements & time lines. Install multilevel contacts between teams because, all together, they deliver results and therefore must be fully engaged in the process. They must understand that they "share the gain", and also "share the pain". On the other hand, it is also very important to create an environment where it is okay to make mistakes without the fear of consequences. Otherwise, rigorous controls will lead to a no-confidence work environment and, in the end, bad results.

So, plan the projects ahead if you want to make them a success. Be aware of all different

scenarios that may arise and yes, have as “elastic” a mindset as possible. And, of course, don’t try to take shortcuts. With this kind of systematic approach, you can smoothly solve internal sourcing issues while releasing complete software solutions within budget, on time and at the right quality. We can proudly confirm that, when it comes to outsourcing, the following statement says it all: ‘the proof of the pudding is in the eating!’

Biography

Aleksandra Popara is Communications Manager in EXECOM, software engineering company and IT services provider. Aleksandra has been working in the software industry for five years covering different positions. She first started as a junior consultant, then joined the system analysis team and rounded off her experience in her recent position in the Marketing & Sales team. Another role Aleksandra holds in EXECOM is Quality Management Officer, responsible for ISO 9001:2000 related processes.

Predrag Skokovic has more than 10 years of experience in IT business. His first job, after becoming a bachelor of computer science, was as a System Administrator. He was introduced to software testing at Execom. Currently he holds a position of the leading Test Manager. Also, he is a member of the Quality Management team. Occasionally, as requested, he gives presentations/ tutorials at the local university about software testing.

ISTQB® Certified Tester Foundation Level

3 days course in **Palma de Mallorca**

May 20 - 22, 2009

Day 1: 10:00-18:00 hours
Day 2: 9:00-17:00 hours
Day 3: 9:00 -15:30 hours (Exam 16:00-17:00 hours)

OBJECTIVES

The seminar Certified Tester – Foundation Level conveys the most relevant testing procedures and testing methods which enable an effective and efficient preparation and execution of software tests. After successful completion of the seminar, you will be able to apply the techniques you learned and make a decisive contribution to the success of your projects.

CONTENTS

The seminar follows the syllabus of the International Software Testing Qualifications Board (ISTQB) and handles the following subjects:

- **Fundamentals of testing**
- **Testing throughout the software life cycle**
- **Static techniques**
- **Dynamic techniques**
- **Techniques for designing test cases**
- **Test management**
- **Tool support for testing**

SEMINAR SERVICES

The price for the seminar includes the following services:
- **Execution of the seminar with a maximum of 10 participants**
- **Snacks and lunch**
- **A printed set of the slides and material**

Subsequent to the seminar, the ISTQB® exam will be offered for those who wish to acquire the certification "Certified Tester – Foundation Level". The exam expenses of 200,- EUR plus Tax per participant will be invoiced from the examination body separately.

PREREQUISITES

Basic practical experience in IT projects and elementary know how of software development are expected. Initial practical experience in software testing is helpful but not required.

TARGET AUDIENCE

Project managers
Quality managers
Software testers
Software developers

1450,- EUR

(plus VAT)

http://training.diazhilterscheid.com/
Register at training@diazhilterscheid.com

Díaz Hilterscheid

© iStockphoto.com/goodynewshoes

To be or not to be

(a company who chooses to outsource software testing)

by Todd E. Sheppard

On the surface the choice to outsource software testing in many cases may seem like an easy decision to make. I can save how much by not having testers in house? Let's do it! In reality the choice is never that easy and rarely based solely on cost. There are many other factors that must be considered when deciding whether or not to outsource software testing. This article explores a few of these factors.

Is the software development outsourced?

If the software development work has been outsourced this may be a good reason for the testing to remain in house. For software applications to be successful they must meet the needs of the user group. These needs should have been clearly defined by a Business Analyst who is in frequent contact with the end users. These needs or requirements are then transmitted to the outsourced software development team. In order for the company to know that the application development team is fulfilling their contract there is a need to have a solid software testing group in house who can evaluate the software as it is being presented for payment. The testing group has access to the original requirements, which should have been met, as well as the business knowledge to validate that the product will fulfill the ultimate needs of the business. The testing group will also validate that the software complies with usability and other internal company standards, which were transmitted along with the requirements.

Is the software being tested for purely internal use or is it customer facing?

While it should never be acceptable to sign off on software that has not been thoroughly tested, the ultimate audience of the software may raise the stakes considerably. If a piece of software that is developed for internal use is deployed in a poor state, this is a waste of company resources. If that same piece of software is destined for deployment to your customers this could be a public relations nightmare or, worse yet, may cost you customers. A poorly functioning software interface, website, or application can add that little extra bit of frustration that drives your customer to a competitor. Once that customer is gone, the cost of lost business may easily outweigh any savings gleamed from outsourcing.

How large is your companies IT department?

If the size of your IT department is small or if the number of applications which you develop is not large enough to justify a full time testing resource, your choices are limited. Outsourcing software testing may be a good option. By outsourcing you have an external resource performing true independent verification. The outsourced tester may have a better knowledge of user interfaces, testing automation tools, requirement validation, and a generally better aptitude for discovering software issues than any of your staff resources. The other option would be to have the developer or end user perform validation testing. Both of these options present problems. It is difficult for the developer to be objective when evaluating their own software. If the software tester is the end user, you run the risk of new "requirements" being introduced during testing, or perhaps worse, exposing the user to the software too early such that they determine it to be too difficult to use due to the issues, thereby reducing their support for the software. On the other hand it can also be a very positive experience to have the end user involved in testing. By bringing in the high end user to help with testing they can help craft the software to meet their needs, be ready pre-deployment to train other users, and provide the critical buy-in when representing the project to upper management.

Is the software being developed an upgrade?

If the software being developed is an upgrade to an existing system which has been previously documented it may be a good candidate for outsourcing. Existing systems being upgraded which have a good amount of documentation, especially if that documentation includes test scripts, are generally easier to hand over to an outside group to perform testing. If the system is one which is likely to need testing again in the future, now may be the time to outsource the testing in an effort to have the testing automated, thereby speeding up future system testing.

What is the exposure to legal liability?

Does this piece of software control data that could present a potential legal liability to the company? For example, there are many software systems that interface with various governmental agencies. Defects in these systems can result in expensive fines or a potential halting of business. The potential legal implications must be considered when determining if outsourcing of testing should be pursued. Is there value in hiring in an outside company that has experience dealing with the government entity? Does the outsourcing company accept responsibility for any defects, legal proceedings and fines, or is your company still responsible? Are there experienced internal testers available or is there time to train new testers if none are available?

Is the function being tested a generalized function?

There are many types of tests that can be outsourced due to their simplicity, allowing inter-

nal testers to focus more on business specific development. The testing of general functions such as Multiple Language testing, common accounting functions, or a simple data input screen could be easily documented and handed out for testing. This brings up another question; why are you developing software which could easily be acquired off of the shelf? That will have to wait for another article though.

Does this application contain proprietary information?

Systems are often developed which contain sensitive information. It is simple enough to translate the data before dispatching it for testing, but it is also simple to inadvertently transmit real data to the testing company. Confidentiality agreements, (you did remember to draft those when hiring a testing firm didn't you?), should protect you to some degree. If the data or software contains trade secrets or by early implementation provides a competitive advantage, it may be prudent to maintain complete control of the software.

Do you have a committed liaison?

The software has been shipped to the outsourced testing company, now you just sit back and wait for the results, right? Not if you hope to successfully deploy the software in any reasonable time frame. A liaison between the testing firm and the company is an absolute necessity. The liaison is there to validate that the defects identified are properly documented, to ensure the testing reports are clear, to manage the contract, and to provide an interface to the development group as needed.

Do you have a budget for test equipment?

Many systems run on rather costly hardware. It is not uncommon to have two or more sets of hardware for large systems. One (or two if redundancy is required) sets for production, and another set for the development team. It is possible to test on the development equipment, but this does bring about some difficulties. If testing is run on the development equipment and the developers are also using it to perform defect correction, this may require large periods of tester downtime. Testing on the development equipment usually also makes it difficult to test any installation procedures as the code is already deployed. It may not be feasible to purchase an additional set of hardware for testing. If the correct outsourcing company is selected they may be able to split the cost of dedicated test equipment over all of their clients.

To outsource or not to outsource?

So which is it to outsource or not to outsource? By outsourcing testing it is possible to utilize highly trained testers without having them constantly on the payroll. You may be looking to purchasing expertise in testing tools that can speed up the test execution. The need to purchase additional hardware to be used for system testing could be eliminated as the outsourcing company may provide the required equipment. Outsourcing does come with its own set of challenges. The liaison to the testers is a crucial role. The liaison must have a solid understanding of software testing, proper documentation, and the business needs. To outsource testing the documentation must be in top shape. If the requirements or system documentation are not clear, the outsourced testing may be of limited value. If portions of the system are not well documented, the testers may "pass" code which does not meet the users' needs and "fail" code which is working perfectly well, but was not documented properly. This would be a waste of not only the consulted company time (your money), but is also wasting the time of your liaison and development team. To break into outsourcing an "ease into it" approach may be the most effective, time permitting. Identify your liaison, identify the company to which you would like to outsource, define a small portion of an application and execute a contract. During this small test go through the effort to define all of the processes which will be necessary for a larger contract including, but not limited to: defect reporting, status reporting, testing timelines, fees, and product liability. Once complete, perform a cost/benefit analysis that includes the quality of the work received, and if this trial is successful consider expanding the contract.

This is all as opposed to training and maintaining an internal testing group. Used correctly internal testers can be an invaluable resource. Internal testers understand the business, understand why the software is being developed, and may be able to offer clues to future enhancements to provide value. A software testing team should be engaged in the whole product lifecycle. The tester can be there when the business requirements are being discussed, ensuring that the requirements as documented are testable. A true quality assurance person can look at the process and offer recommendations for improvement, whether it is writing better requirements, changing a business process, or more clearly documenting defects. While software development is under way internal testers can be documenting test scripts and plans, thinking through the testing. This often yields "paper defects" which can be addressed before development is complete by raising the issue early. Hiring an outside testing group to perform this level of service for the duration of a project could be quite a costly proposition. With the proper vision an internal testing team will span multiple platforms, multiple projects, and be agile enough to respond to changes as they arise. It is much easier to re-plan work for an internal team than to renegotiate a contract with an external group, if shifting business realities make this a necessity. When tests are automated, the maintenance is often costly, but an internal resource actively maintaining the automation scripts as opposed to hiring an external company to relearn and then update the scripts can be a cost advantage.

There is no simple answer as to when, what, or how much software testing should be outsourced. There are situations that will call for all internal testers, all external testing, or a mix of both. Each case needs to be individually evaluated and, using all of the knowledge available, the correct decision to support the business needs to be made. In the end, as testers or testing managers, our job is to ensure the organization receives quality software as economically as possible, no matter who ends up performing the testing.

Biography

Todd E. Sheppard has been working as an IT professional for the past 16 years. Todd holds a Master of Engineering degree from the University of Louisville, Kentucky USA. He has worked as a System Integrator, a Software Developer, in Training and Deployment, a Release Manager, and a Quality Assurance Analyst. Todd has been focused solely on software testing and quality assurance for the past 9 years. He is currently employed as a Sr. Quality Assurance Analyst at UPS.

Intelligent Use of Testing Service Providers

by Rex Black

This article is excerpted from Chapter 10 of Rex Black's upcoming book Managing the Testing Process, 3e.

Should we outsource our testing? What testing tasks should we outsource, and what tasks should we keep in-house? What are the chief advantages and disadvantages of outsourcing testing? As a consultant, I've been asked questions like these and other outsourcing-related questions a number of times. For a few clients, I've provided the following analysis, which should prove useful to you if you are evaluating outsourcing of testing.

In this article, I analyze the use of outsourcing in testing, based on some twenty years of experience with outsourcing of testing in one form or another. First, I enumerate the key differences between in-house and outsourced test teams. Next, driven by these key differences, I'll analyze which tasks fit better with outsourced testing service providers, followed by a similar analysis for in-house test teams. Then, I'll list some of the technical, managerial, and political challenges that confront a company trying to make effective use of outsourced testing. Finally, I'll address some of the processes needed to use testing service providers effectively and with the least amount of trouble.

Key Differences between Testing Service Providers and In-house Test Teams

A number of important differences exist between testing by an in-house test team and testing by a testing service provider. I want to start by listing these here, because I'll refer to them later in this analysis (by the labels *KD1, KD2*, etc.). The key differences include, most especially, the following:

KD1. Testing service providers already exist. An organization can use them almost immediately for any project. Building an in-house test team can take longer than the duration of a project.

KD2. The cost structure of a testing service provider is variable, while an in-house test team has start-up costs (infrastructure purchase), fixed costs (both labor and infrastructure maintenance), and variable costs (generally project-related contract labor).

KD3. Testing service providers spread the purchase and maintenance costs of particularly expensive infrastructures, such as network test facilities and extensive hardware/software libraries, across a large number of (customer) organizations. An in-house test team must incorporate these costs into the development budget.

KD4. Testing service providers often have existing documented test cases, suites, procedures and processes that have proven efficient (based on the test labs' continued existence). An in-house test team may not have the competitive pressures and sufficient repetition required to hone such tools.

KD5. When the testing service provider is focused entirely on testing, it will be typically staffed by professional testers, people who choose to work in the test field. (Note that you should carefully scrutinize the test staffs of larger outsourcing firms that include a so-called testing center of excellence or testing practice before concluding that they consist of test professionals.) In-house test teams often attract people who want to move into other areas in the organization.

KD6. In some cases, two or more competitors use the same testing service providers. An in-house test team can provide unique, focused services with no risk of intellectual property leakage.

KD7. Testing service providers tend to run lean. An in-house test team may fully staff a project to avoid temporary stalls in project progress.

KD8. Testing service providers are typically off-site (though some provide the services in part or in whole through on-site staff augmentation). In-house test teams are more likely to be collocated with the engineering and development facilities.

KD9. Testing service providers are most economical when used in a single-pass mode (i.e., once through the planned test suite). They are often uneconomical when used in cyclic activities, in comparison to an in-house test team.

With these differences in mind, let's now analyze how these differences affect outsourced testing.

Test Tasks Appropriate for Testing Service Providers

Now, let's look at how these key differences make certain tasks particularly appropriate for outsourcing to testing service providers. First, due to the ready, willing and able nature of external labs (KD1), test projects which arise suddenly, contain spikes in personnel needs, or exceed the capacity of the current internal test organization are appropriate for outsourcing to a testing service provider. Also, the variable cost structure (KD2) implies that, even in cases where one could staff to meet a spike in test needs, the fixed costs associated with staffing up, plus the variable costs incurred during the testing, might exceed the variable cost incurred by using an external lab instead. So, RBCS recommends the use of testing service providers in most cases of atypical peaks in needed test capacity, rather than staffing to handle the peak, then eliminating staff after the peak.

Diffusing infrastructure costs across multiple customers (KD3) means that, when one needs to test with particularly expensive hardware, or simply a wide range of hardware or software, using a testing service provider makes sense. Network tests are a good example, because, across four or more network topologies (some with three or four protocols), six or seven server types, ten or more client stacks, and almost innumerable client network adapter options, the level of complexity and the fixed and variable costs associated with such a lab are high. Such labs require full-time network administrators who command large salaries. Also, testing service providers maintain extensive libraries of software and hardware, having full-time acquisitions managers who keep the test tools current. Such expenses are impractical for small to mid-sized companies, yet compatibility and configuration testing are essential in many mass-market and Internet products and services. RBCS recommends the use of testing service providers to provide broad test coverage in these areas as a particularly smart reason to hire a testing service provider.

Efficient test processes and expert test professionals (KD4 and KD5) are especially useful in areas of testing considered unglamorous. Testing service providers generally hire and retain a greater proportion of professional testers than do hardware, software, and systems companies staffing in-house test teams, and certainly more than the average non-computer-related company hiring testers for their IT operation. In-house test team candidates often have long-term goals in other fields within the company, viewing testing as a stepping stone to advancement. In-house test team staff members usually enjoy testing cutting-edge technologies, and developing test plans for products that use these technologies. When it comes to more mundane but equally important tasks, testing service providers will usually have an edge.

The remote location of the testing service provider (KD8), while sometimes perceived as a disadvantage, can provide benefits. First, it offers an environment where the test effort can stay focused on the overall test plan, rather than getting bogged down in the crisis of the moment. Second, in conjunction with a dedicated liaison, a single contact window provides a way of channeling information from the test staff to the project team without non-test team members interrupting testers with what they think will be (but often are not) quick questions, a significant time drain for in-house test teams. This means that one can hand off the bulk of the test tasks to a testing service provider, knowing they will proceed towards the agreed-upon goals for completion in a deliberate fashion, regardless of the tempests raging around the project on-site.

Test Tasks Appropriate for In-house Test Teams

Now, let's look at how the key differences make certain tasks particularly appropriate to retain for an in-house test team. First, a two-edged sword, the variable cost structure of external labs (KD2) implies that you need to use testing service providers carefully, and with forethought. Though the cost for a single test may be relatively low, the cost per test cycle (one complete run through all planned tests) might not be. Therefore, you need to keep the number of cycles executed at an external lab to a minimum in many cases. Handling crisis situations, especially those sparked by last-minute changes in requirements, is better suited to an in-house test team, at least at first, testing service provider responsiveness (KD1) and leveraged infrastructure (KD3) notwithstanding. In some cases, of course, the infrastructure available at the testing service provider is necessary (e.g., a complex network or a range of supported browsers). If testing service provider participation in a crisis is needed, the test manager must take time to plan carefully their role. Otherwise, you might incur excessive cycles at the testing service provider, and you'll typically pay for these services on a time-and-materials basis.

While documented, standard test processes (KD4) would suggest that testing service providers will be more outperform than in-house test teams, this often is most true for tasks which must be done efficiently (i.e., with the minimum cost and effort). Sometimes the most important consideration is completeness of coverage and meticulousness of testing, especially for those features and functions judged by business stakeholders as critical. Also, the broad client base and cross-platform service offering of testing service providers (KD6) means solid testing and technology expertise, they tend to be weak in highly specialized application domains, since they often cannot afford to specialize in the features, advanced functions, and peculiarities applicable to a single client's system. These two facts have a number of implications. First, some level of in-house test expertise is needed to make the decision on whether bugs reported by testing service providers have the appropriate severity and priority assigned, and, indeed, whether they are bugs at all, given the unique elements of the system. Second, in-house test staff will need to cover the unusual features in the system, or be prepared to provide direct support to the testing service provider in terms of what testing should occur. Third, in-house test staff will need to test the most critical features, even though you might also have the external labs provide double coverage for added security.

In a mass-market, e-commerce, or software-as-a-service situation, where many of your competitors often substantially similar products and services, the openness of the external labs (KD6) means that you can't obtain a significant competitive edge in terms of testing and quality by using testing service providers exclusively. While using a dedicated testing service provider liaison allows for a great degree of specialization and more-complete coverage, to achieve truly distinctive levels of quality, an organization must add testing in an in-house test team. Furthermore, some of the testing done by this team must utilize the information available from business stakeholders and others with direct interaction with customers and users to emulate the real customers and users in terms of infrastructure environments, test scenarios and usage profiles. In other words, while in-house test teams have the potential to provide a competitive edge in testing, that edge does not come free, but rather requires considerable forethought, cooperation and hard work.

The lean staffing of testing service providers (KD7) affects handling of the critical test tasks, those in which speed of completion, accuracy of results, and thoroughness are all of crucial importance. A good example is performance testing. In most testing service providers, specialized personnel are not always available to work on your particular crisis task at the exact moment you require the service, and you or others in your organization might find it unacceptable to have the testing service provider place such crisis tasks in a queue for service based on specialist availability. (Of course, the distinction here may well be one of appearance – the perception of a uniform sense of urgency – as any actual difference in the time required to complete the task. A testing service provider might, through efficiency and expertise, hold an edge in handling such tasks, but the politics of the crisis might require someone present on-site, visibly sweating the details and working long hours.) Note that location (KD8) comes into play here as well, particularly when the priorities tend to change a few times a day, with the latency of communication to an off-site testing service provider making it difficult for them to keep up with rapidly evolving conditions.

The rapidly mounting costs of external labs in a cyclic mode (KD9) affects test tasks such as regression testing. While using a testing service provider to provide a final check on the product makes sense, delivering highly buggy products for testing can result in multiple passes through the test suite at considerable

A Communit

100,000 Ce

Are you on th

www.is

*October 2008


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

expense. An in-house test team should ensure sufficient quality prior to engagement of a testing service provider, using written entry criteria.

Organizational Challenges

Using a testing service provider can involve resolving a number of technical, managerial and political challenges in many organizations. Many of these challenges apply to distributed testing in general; e.g., leveraging a vendor's test organization. Let's take a look at these challenges so that you can resolve them before they become obstacles.

First, since there is no widely-accepted definition of a test case, each test organization involved will have cases consisting of a different number of conditions (granularity) and requiring a different amount of time. Also, some tests naturally break down into different sized tasks. The test manager must ensure that they have a way to integrate tests and test information across disparate testing teams in a way that makes sense.

In addition to test case definitions, a continuum also exists in terms of where debugging tasks end and where testing tasks begin. This is multidimensional, applying to the responsibilities of developers versus testers, when testers have performed sufficient isolation to characterize a bug in a report sufficient for delivery to the development team, and when developers have performed sufficient testing to return a fix to the test team. Each organization will have a different answer. This is an issue for testing service providers in that you need an agreement in advance on the level of isolation performed for each bug report. Failure to get sufficient isolation information in the bug reports from the testing service providers will result in a lot of questions and complaints by developers about bugs reported by the testing service provider.

In development projects that have international teams, language will be an issue. You cannot assume that all testers will know the project language, nor can you assume that a project team member from another country who is conversant in the project language will read and write it as effortlessly as a native speaker. The organization must make allowances for varying levels of language proficiency, and plans must exist for translating critical documents into the project language.

Influenced by the three issues above but distinct from them is the matter of test result reporting. While there is a certain minimal set of data one expects to find in a bug report, standards and formats vary widely. In addition, the ways in which organizations track the status (pass, fail, blocked, etc.) of test cases vary both in method and in quality. The overall test results dashboard is likely to vary from one organization to another. While converting everyone to a single format is unlikely, crucial milestones prior to starting test execution include agreeing on a minimal set of data, a frequency of reporting, standards for failure reproduction procedures, and standards for updating changing status. In addition, the test manager will need to create a single, unified test results dashboard for reporting to project and outside managers, with quick, meaningful data feeds across all the test teams, both internal and external.

Testing should be pervasive, in the sense that there are a series of levels of testing and quality assurance activities embedded in the project. Each level should have established entry and exit criteria. This allows each activity to act as a bug-filter, leading to the highest cumulative level of bug removal in the most efficient fashion. In contrast, a single-level approach with no formal entry or exit criteria tends to rush products into testing before they are ready, to squander a lot of time in testing trying to test unready, unstable products, and to release systems with low and inconsistent levels of quality. However, while this multi-filter approach is well-established in mature organizations, and general guidelines for test levels are available, organizations apply them differently. This means that a vendor might use A-Test, B-Test, and C-Test, your company might use Component Test, Integration Test, and System Test, and the testing service provider might use a whole different standard. Alternatively, they might all use the same names for the levels but have different objectives, entry criteria, and exit criteria for them. The manager responsible for the master test plan on a given project must understand how all these levels fit together; otherwise, the benefits of the test levels will be defeated.

On systems projects that involve hardware prototypes, prototype allocation will usually create a headache or two. You should pay attention to this issue both during test planning and during test control. Because more organizations are involved, you'll need more prototypes. Before test execution starts, you should make sure that an agreed-upon allocation plan exists. This plan should, to the extent possible, meet the needs of all test participants. The compromises made – and you will make compromises – due to system scarcity should be understood and clearly communicated. During test execution, you must set and monitor milestones to ensure that allocated quantities are delivered on time. You should proactively anticipate and handle any delays and shortages that might occur.

With downsizing always a possibility, and leading a large department sometimes seen as part of climbing the management ladder, the use of testing service providers for reasons other than pure staffing necessity tends to meet with some resistance from the in-house test team. This is less of a challenge for an organization planning to use testing service providers from the beginning, than for organizations with large, existing in-house test teams that are starting to use test outsourcing. However, the political pressures exist to some degree regardless, and the organization must deal with these pressures effectively to prevent dysfunctional behavior.

The first five challenges, by the way, come under a general area called mapping. They involve taking the test processes of different teams and mapping them functionally into a single virtual test team. This is, of course, less efficient than having a single set of standards to which all test teams adhere from the start, but, in the case of outside testing service providers and vendors, there is no real alternative.

Processes for Effective Use of Testing Service Providers

Related to the issue of mapping is alignment of processes. While you cannot – and need not – resolve all such process alignment issues to the point where everyone involved in testing works the same as your in-house test team does, you need to ensure good process alignment in the following areas to support effective use of testing service providers.

- Someone must own the role of test coordinator or liaison to ensure integration of the test results into the test management and reporting processes.
- Processes must exist to deal with the mapping issues that arise.
- During the planning phases for the overall test project, a global plan should identify all the test work that will occur, regardless of location. The test manager should consider the testing service providers as available testing resources, and plan accordingly.
- Since no testing service provider works for free, a way of getting them paid must exist.
- Though e-mail and phone calls can handle much of the coordinating of external testing, nothing maximizes communication effectiveness like physical presence, especially in international situations where project language issues exist. Therefore, the coordinator or liaison role must involve travel to the testing service provider's location as required. In addition, this liaison should make sure that developers address the bugs found and reported by the testing service providers.
- Whenever hardware prototypes will remain on-site at a testing service provider for any length of time, a process must exist for refreshing the prototypes as needed. Allowing the prototypes to get out of date in terms of firmware and hardware versions in comparison to the final hardware will cause test escapes.

All these processes must include active follow-up elements for the coordinator, liaison, and/or test manager. In any distributed project, especially one that spans time zones, multiple opportunities exist for confusion, miscommunication, lost communication and ostrich-like reactions to bad news or undesirable requests. While distributed testing creates a matrix-like test organization, the ultimate responsibility for test task completion must remain with one person.

Conclusion

Properly used, testing service providers can make a valuable and economical contribution to system and software projects. Categories of tests that make sense to outsource include:

- Tests that would cause a short-term spike in test manpower requirements, followed by downtime or layoffs.
- Tests that cover a board range of hardware and software, or otherwise use expensive infrastructures.
- Tests that are routine and unglamorous, but important to shipping a quality product.
- Large blocks of tests that might not get done in-house due to changing priorities.

Some tests require in-house attention. Categories of tests that make sense to run using in-house test teams include:

- Tests that result from last-minute changes in requirements.
- Tests and test result evaluations that require sophisticated domain knowledge.
- Tests that provide a competitive advantage in terms of evaluating and improving product quality.
- Tests that are time-critical and vital to project success, or which are subject to rapidly evolving requirements.
- Tests that establish system stability prior to release to an testing service provider for final testing.

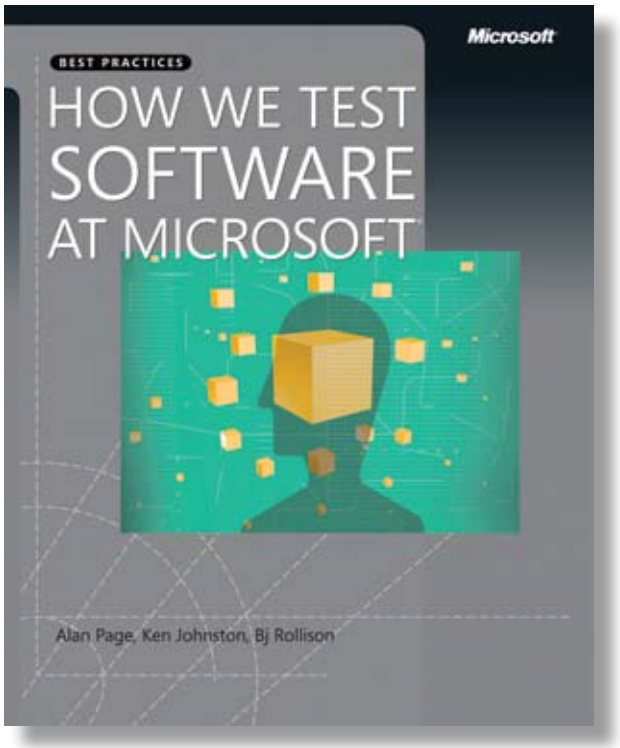
In addition to dividing the test workload intelligently, the effective test manager must handle the technical, managerial and political issues associated with using external and distributed test resources. Many of these problems have to do with mapping and integrating the disparate test organizations into a single virtual test team. In addition, the test manager must allocate test units effectively and make intelligent allocation compromises as needed. Finally, once the test manager has the work divided and the virtual organization mapped, the test manager must ensure that appropriate processes exists to support the execution of the project. These processes must support a single point of contact for gathering test results into a consistent test dashboard and to ensure completion of critical tasks.

Biography

With a quarter-century of software and systems engineering experience, Rex Black is President of RBCS (www.rbcs-us.com), a leader in software, hardware, and systems testing. For over a dozen years, RBCS has delivered services in consulting, outsourcing and training for software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS conducts product testing, builds and improves testing groups and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. As the leader of RBCS, Rex is the most prolific author practicing in the field of software testing today. His popular first book, Managing the Testing Process, has sold over 35,000 copies around the world, including Japanese, Chinese, and Indian releases. His five other books on testing, Advanced Software Testing: Volume I, Advanced Software Testing: Volume II, Critical Testing Processes, Foundations of Software Testing, and Pragmatic Software Testing, have also sold tens of thousands of copies, including Hebrew, Indian, Chinese, Japanese and Russian editions. He has written over twenty-five articles, presented hundreds of papers, workshops, and seminars, and given about thirty keynote speeches at conferences and events around the world.
Rex is the President of the International Software Testing Qualifications Board and a Director of the American Software Testing Qualifications Board.

INTRODUCING

- Discover how Microsoft implements and manages the software-testing process company-wide
- Gain expert insights on effective testing techniques and methodologies
- Shares such facts as the number of test machines at Microsoft, how the company uses automated test cases, and bug statistics
- Answers key testing questions, such as who tests what, when, and with what tools
- Describes how test teams are organized, when and how testing gets automated, testing tools, and feedback



HOW WE TEST SOFTWARE AT MICROSOFT
ISBN: 9780735624252

ABOUT THE AUTHORS

Alan Page has been a software tester for more than 14 years, including more than 12 years at Microsoft Corporation.
Ken Johnston is group manager for testing in the Microsoft Office business group.
Bj Rollison is a Test Architect in the test excellence organization at Microsoft.
Authors' website: http://www.hwtsam.com

Microsoft® Press

Buy the book! **http://www.microsoft.com/learning/en/us/books/11240.aspx**

What happens to usability when development goes offshore?

by James Christie

Usability fails to keep pace with surge in offshoring

Two of the most important trends in software development over the last 20 years have been the increasing number of companies sending development work to cheaper labour markets, and the increasing attention that is paid to the usability of applications.

Developers in Europe and North America cannot fail to have missed the trend to offshore development work and they worry about the long-term implications.

Usability, however, has had a mixed history. Many organizations and IT professionals have been deeply influenced by the need to ensure that their products and applications are usable; many more are only vaguely aware of this trend and do not take it seriously.

As a result, many developers and testers have missed the significant implications that offshoring has for building usable applications, and underestimate the problems of testing for usability. I will try to explain these problems, and suggest possible remedial actions that testers can take if they find themselves working with offshore developers. I will be looking mainly at India, the largest offshoring destination, because information has been more readily available. However, the problems and lessons apply equally to other offshoring countries.

According to NASSCOM, the main trade body representing the Indian IT industry [1], the numbers of IT professionals employed in India rose from 430,000 in 2001 to 2,010,000 in 2008. The numbers employed by offshore IT companies rose 10 fold, from 70,000 to 700,000.

It is hard to say how many of these are usability professionals. Certainly at the start of the decade there were only a handful in India. Now, according to Jhumkee Iyengar, of User in Design, "*a guesstimate would be around 5,000 to 8,000*". At most that's about 0.4% of the people working in IT. Even if they were all involved in offshore work they would represent no more than 1% of the total.

Does that matter? Jakob Nielsen, the leading usability expert, would argue that it does. His rule of thumb [2] is that "*10% of project resources must be devoted to usability to ensure a good quality product*". Clearly India is nowhere near capable of meeting that figure. To be fair, the same can be said of the rest of the world given that 10% represents Nielsen's idea of good practice, and most organizations have not reached that stage. Further, India traditionally provided development of "back-office" applications, which are less likely to have significant user interaction.

Nevertheless, the shortage of usability professionals in the offshoring destinations does matter. Increasingly offshore developments have involved greater levels of user interaction, and any shortage of usability expertise in India will damage the quality of products.

Sending development work offshore always introduces management and communication problems. Outsourcing development, even within the same country, poses problems for usability. When the development is both offshored and outsourced, the difficulties in producing a usable application multiply. If there are no usability professionals on hand, the danger is that the developers will not only fail to resolve those problems - they will probably not even recognize that they exist.

Why can outsourcing be a problem for usability?

External software developers are subject to different pressures from internal developers, and this can lead to poorer usability. I believe that external suppliers are less likely to be able to respond to the users' real needs, and research supports this. [3, 4, 5, 6, 7]

Obviously external suppliers have different goals from internal developers. Their job is to deliver an application that meets the terms of the contract and makes a profit in doing so. Requirements that are missed or are vague are unlikely to be met, and usability requirements all too often fall into one of these categories. This is not simply a matter of a lack of awareness. Usability is a subjective matter, and it is difficult to specify precise, objective, measurable and testable requirements. Indeed, trying too hard to do so can be counter-productive if the resulting requirements are too prescriptive and constrain the design.

A further problem is that the formal nature of contractual relationships tends to push clients towards more traditional, less responsive and less iterative development processes, with damaging results for usability. If users and developers are based in different offices, working for different employers, then rapid informal feedback becomes difficult.

Some of the studies that found these problems date back to the mid 90s. However, they contain lessons that remain relevant now. Many organizations have still not taken these lessons on board, and they are therefore facing the same problems that others confronted 10 or even 20 years ago.

How can offshoring make usability problems worse?

So, if simple outsourcing to a supplier in the same country can be fraught with difficulty, what are the usability problems that organizations face when they offshore?

Much of the discussion of this harks back to an article by Jakob Nielsen in 2002 [2]. Nielsen stirred up plenty of discussion about the problem, much of it critical.

"*Offshore design raises the deeper problem of separating interaction designers and usability professionals from the users. User-centered design requires frequent access to users: the more frequent the better.*"

If the usability professionals need to be close to the users, can they stay onshore and concentrate on the design while the developers build offshore? Nielsen was emphatic on that point. "*It is obviously not a solution to separate design and implementation since all experience shows that design, usability, and implementers need to proceed in tight co-ordination. Even separating teams across different floors in the same building lowers the quality of the resulting product (for example, because there will be less informal discussions about interpretations of the design spec).*"

So, according to Nielsen, the usability professionals have to follow the developers offshore. However, as we've seen, the offshore countries have nowhere near enough trained professionals to cover the work. Numbers are increasing, but not by enough to keep pace with the growth in offshore development, never mind the demands of local commerce.

This apparent conundrum has been dismissed by many people who have pointed out, correctly, that offshoring is not an "all or nothing" choice. Usability does not have to follow development. If usability is a concern, then user design work can be retained onshore, and usability expertise can be deployed in both countries. This is true, but it is a rather unfair criticism of Nielsen's arguments. The problem he describes is real enough. The fact that it can be mitigated by careful planning certainly does not mean the problem can be ignored.

User-centred design assumes that developers, usability analysts and users will be working closely together. Offshoring the developers forces organizations to make a choice between two unattractive options; separating usability professionals from the users, or separating them from the developers.

It is important that organizations acknowledge this dilemma and make the choice explicitly, based on their needs and their market. Every responsible usability professional would be keenly aware that their geographical separation from their users was a problem, and so those organizations that hire usability expertise offshore are at least implicitly acknowledging the problems caused by offshoring. My concern is for those organizations who keep all the usability professionals onshore and either ignore the problems, or assume that they don't apply in their case.

How not to tackle the problems

Jhumkee Iyengar has studied the responses of organizations wanting to ensure that offshore development will give them usable applications [8]. Typically they have tried to do so without offshore usability expertise. They have used two techniques sadly familiar to those who have studied usability problems; defining the user interaction requirements up-front and sending a final, frozen specification to the offshore developers, or adopting the flawed and fallacious layering approach.

Attempting to define detailed up-front requirements is anathema to good user-centred design. It is an approach consistent with the Waterfall approach and is attractive because it is neat and easy to manage (as I discussed in my article on the V Model in Testing Experience, issue 4). Building a usable application that allows users and customers to achieve their personal and corporate goals painlessly and efficiently requires iteration, prototyping and user involvement that is both early in the lifecycle and repeated throughout it.

The layering approach was based on the fallacy that the user interface could be separated from the functionality of the application, and that each could be developed separately. This fallacy was very popular in the 80s and 90s. Its influence has persisted, not because it is valid, but because it lends an air of spurious respectability to what people want to do anyway. Academics expended a huge amount of effort trying to justify this separability. Their arguments, their motives and the consequences of their mistake are worth a full article in their own right. I'll restrict myself to saying that the notion of separability was flawed on three counts.

- It was flawed conceptually because usability is a product of the experience of the user with the whole application, not just the interface.
- It was flawed architecturally, because design decisions taken by system architects can have a huge impact on the user experience.
- Finally, it was flawed in political and organizational terms because it encouraged usability professionals to retreat into a ghetto, isolated from the hubbub of the developers, where they would work away on an interface that could be bolted onto the application in time for user testing.

Lewis & Rieman [3] memorably savaged the idea that usability professionals could hold themselves aloof from the application design, calling it "*the peanut butter theory of usability, in which usability is seen as a spread that can be smeared over any design, however dreadful, with good results if the spread is thick enough. If the underlying functionality is confusing, then spread a graphical user interface on it. ... If the user interface still has some problems, smear some manuals over it. If the manuals are still deficient, smear on some training which you force users to take.*"

If the usability professionals stay onshore, and take either of these approaches, the almost inescapable result is that they will be delegating decisions about usability to the offshore developers. Developers are just about the worst people to take these decisions. They are too close to the application, and instinctively see workarounds to problems that might appear insoluble to real users. They also have a different mindset when approaching technology. Even if they understand the business context of the applications they can't unlearn their technical knowledge and experience to see the application as a user would; and this is if developers and users are in the same country. The cultural differences are magnified if they are based in different continents.

The relative lack of maturity of usability in the offshoring destinations means that developers often have an even less sophisticated approach than developers in the client countries. User interaction is regarded as an aesthetic matter restricted to the interface, with the developers more interested in the guts of the application.

Pradeep Henry reported in 2003 that most user interfaces at Indian companies were being designed by programmers, and that in his opinion they had great difficulty switching from their normal technical, system-focused mindset to that of a user. [9]

They also had very little knowledge of user centered design techniques. This is partly a matter of education, but there is more to it. In explaining the shortage of usability expertise in India, Jhumkee Iyengar told me that she believes important factors are the "*phenomenal success of Indian IT industry, which leads people to question the need for usability, and the offshore culture, which has traditionally been a 'back office culture' not conducive to usability*".

The situation is, however, certainly improving. Although the explosive growth in development work in India, China and Eastern Europe has left the usability profession struggling to keep up, the number of usability experts has grown enormously over the last 10 years. There are nowhere near enough, but there are firms offering this specialist service keen to work with offshoring clients. This trend is certain to continue because usability is a high value service. It is a hugely attractive route to follow for these offshore destinations, and complementing and enhancing the traditional offshore development service.

Testers must warn of the dangers

The significance of all this from the perspective of testers is that even though usability faces significant threats when development is offshored, there are ways to reduce the dangers and the problems. They cannot be removed entirely, but offshoring offers such cost savings it will continue to grow and it is important that testers working for client companies understand these problems and can anticipate them.

.NET Performance

4 days course by Sasha Goldshtein

April 27 - 30, 2009 in Munich, Germany

Limited places

Description
This four-day instructor-led course provides students with the knowledge and skills to develop high-performance applications with the .NET Framework. Building high-performance applications with the .NET Framework requires deep understanding of .NET memory management (GC), type internals, collection implementation and most importantly - tools for measuring application performance. The course features numerous performance measurement scenarios, optimization tricks, deep focus on .NET internals and high-performance development guidelines.

Intended audience
This course is intended for C# developers with practical experience of at least a year with the .NET framework.

2800,- EUR
(plus VAT)

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

te testing experience
Knowledge Transfer

Testers may not always, or often, be in a position to influence whether usability expertise is hired locally or offshore. However, they can flag up the risks of whatever approach is used, and adopt an appropriate test strategy.

The most obvious danger is if an application has significant interaction with the user and there is no specialist usability expertise on the project. As I said earlier, this could mean that the project abdicates responsibility for crucial usability decisions to the offshore developers.

Testers should try to prevent a scenario where the interface and user interaction are pieced together offshore, and thrown "over the wall" to the users back in the client's country for acceptance testing when it may be too late to fix even serious usability defects.

Is it outside the traditional role of a tester to lobby project management to try and change the structure of the project? Possibly, but if testers can see that the project is going to be run in a way that makes it hard to do their job effectively then I believe they have a responsibility to speak out.

I'm not aware of any studies looking at whether outsourcing contracts (or managed service agreements) are written in such prescriptive detail that they restrict the ability of test managers to tailor their testing strategy to the risks they identify. However, going by my experience and the anecdotal evidence I've heard, this is not an issue. Testing is not usually covered in detail in contracts, thus leaving considerable scope to test managers who are prepared to take the initiative.

Although I've expressed concern about the dangers of relying on a detailed up front specification there is no doubt that if the build is being carried out by offshore developers then they have to be given clear, detailed, unambiguous instructions. The test manager should therefore set a test strategy that allows for significant static testing of the requirements documents. These should be shaped by walkthroughs and inspections to check that the usability requirements are present, complete, stated in sufficient detail to be testable, yet not defined so precisely that they constrain the design and rule out what might have been perfectly acceptable solutions to the requirements.

Once the offshore developers have been set to work on the specification it is important that there is constant communication with them and ongoing static testing as the design is fleshed out.

Hienadz Drahun leads an offshore interaction design team in Belarus. He stresses the importance of communication. He told me that "communication becomes a crucial point. You need to maintain frequent and direct communication with your development team."

Dave Cronin of the US Cooper usability design consultancy wrote an interesting article about this in 2004, [10].

"We already know that spending the time to holistically define and design a software product dramatically increases the likelihood that you will deliver an effective and pleasurable experience to your customers, and that communication is one of the critical ingredients to this design process. All this appears to be even more true if you decide to have the product built in India or Eastern Europe.

To be absolutely clear, to successfully outsource product development, it is crucial that every aspect of the product be specifically defined, designed and documented. The kind of hand-waving you may be accustomed to when working with familiar and well-informed developers will no longer suffice."

Significantly Cronin did not mention testing anywhere in his article, though he does mention "feedback" during the design process.

The limits of usability testing

One of the classic usability mistakes is to place too much reliance on usability testing. In fact, I've heard it argued that there is no such thing as usability testing. It's a provocative argument, but it has some merit. If usability is dependent only on testing, then it will be left to the end of the development, and serious defects will be discovered too late in the project for them to be fixed.

"*They're only usability problems, the users can work around them*" is the cry from managers under pressure to implement on time. Usability must be incorporated into the design stages, with possible solutions being evaluated and refined. Usability is therefore produced not by testing, but by good design practices.

Pradeep Henry called his new team "Usability Lab" when he introduced usability to Cognizant, the offshore outsourcing company, in India. However, the name and the sight of the testing equipment in the lab encouraged people to equate usability with testing. As Henry explained;

"*Unfortunately, equating usability with testing leads people to believe that programmers or graphic designers should continue to design the user interface and that usability specialists should be consulted only later for testing.*" Henry renamed his team the Cognizant Usability Group (now the Cognizant Usability Center of Excellence). [9]

Tactical improvements testers can make

So if usability evaluation has to be integrated into the whole development process then what can testers actually do in the absence of usability professionals? Obviously it will be difficult, but if iteration is possible during design, and if you can persuade management that there is a real threat to quality then you can certainly make the situation a lot better.

There is a lot of material readily available to guide you. I would suggest the following.

Firstly, Jakob Nielsen's Discount Usability Engineering [11] consists of cheap prototyping (maybe just paper based), heuristic (rule based) evaluation and getting users to talk their way through the application, thinking out loud as they work their way through a scenario.

Steve Krug's "lost our lease" usability testing basically says that any usability testing is better than none, and that quick and crude testing can be both cheap and effective. Krug's focus is more on the management of this approach rather than the testing techniques themselves, so it fits with Nielsen's DUE, rather than being an alternative in my opinion. It's all in his book "Don't make me think". [12]

Lockwood & Constantine's Collaborative Usability Inspections offer a far more formal and rigorous approach, though you may be stretching yourself to take this on without usability professionals. It entails setting up formal walk-throughs of the proposed design, then iteration to remove defects and improve the product. [13, 14, 15]

On a lighter note, Alan Cooper's book "The inmates are running the asylum" [16], is an entertaining rant on the subject. Cooper's solution to the problem is his Interaction Design approach. The essence of this is that software development must include a form of functional analysis that seeks to understand the business problem from the perspective of the users, based on their personal and corporate goals, working through scenarios to understand what they will want to do.

Cooper's Interaction Design strikes a balance between the old, flawed extremes of structured methods (which ignored the individual) and traditional usability (which often paid insufficient attention to the needs of the organization). I recommend this book not because I think that a novice could take this technique on board and make it work, but because it is very readable and might make you question your preconceptions and think about what is desirable and possible.

Longer term improvements

Of course it's possible that you are working for a company that is still in the process of offshoring and where it is still possible to influence the outcome. It is important that the invitation to tender includes a requirement that suppliers can prove expertise and experience in usability engineering. Additionally, the client should expect potential suppliers to show they can satisfy the following three criteria.

- The supplier should have a process or lifecycle model that not only has usability engineering embedded within it but that also demonstrates how the onshore and offshore teams will work together to achieve usability. The process must involve both sides.
- Offshore suppliers have put considerable effort into developing such frameworks. Three examples are Cognizant's "End-to-End UI Process", HFI's "Schaffer-Weinschenk Method™" and Persistent's "Overlap Usability".

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)

2 years subscription

☐ **60,- €**
(plus VAT)

Date

Signature, Company Stamp

- Secondly, suppliers should carry out user testing with users from the country where the application will be used. The cultural differences are too great to use people who happen to be easily available to the offshore developers. Remote testing entails usability experts based in one location conducting tests with users based elsewhere, even another continent. It would probably not be the first choice of most usability professionals, but it is becoming increasingly important. As Jhumkee Iyengar told me it "*may not be the best, but it works and we have had good results. A far cry above no testing.*"
- Finally, suppliers should be willing to send usability professionals to the onshore country for the requirements gathering. This is partly a matter of ensuring that the requirements gathering takes full account of usability principles. It is also necessary so that these usability experts can fully understand the client's business problem and can communicate it to the developers when they return offshore.

It's possible that there may still be people in your organization who are skeptical about the value of usability. There has been a lot of work done on the return on investment that user centered design can bring. It's too big a topic for this article, but a simple internet search on "usabilty" and "roi" will give you plenty of material.

What about the future?

There seems no reason to expect any significant changes in the trends we've seen over the last 10 years. The offshoring countries will continue to produce large numbers of well-educated, technically expert IT professionals. The cost advantage of developing in these countries will continue to attract work there.

Proactive test managers can head off some of the usability problems associated with offshoring. They can help bring about higher quality products even if their employers have not allowed for usability expertise on their projects. However, we should not have unrealistic expectations about what they can achieve. High quality, usable products can only be delivered consistently by organizations that have a commitment to usability and who integrate usability throughout the design process.

Offshoring suppliers will have a huge incentive to keep advancing into user centered design and usability consultancy. The increase in offshore development work creates the need for such services, whilst the specialist and advanced nature of the work gives these suppliers a highly attractive opportunity to move up the value chain, selling more expensive services on the basis of quality rather than cost.

The techniques I have suggested are certainly worthwhile, but they may prove no more than damage limitation. As Hienadz Drahun put it to me; "*to get high design quality you need to have both a good initial design and a good amount of iterative usability evaluation. Iterative testing alone is not able to turn a bad product design into a good one. You need both.*" Testers alone cannot build usability into an application, any more than they can build in quality.

Testers in the client countries will increasingly have to cope with the problems of working with offshore development. It is important that they learn how to work successfully with offshoring and adapt to it. They will therefore have to be vigilant about the risks to usability of offshoring, and advise their employers and clients how testing can best mitigate these risks, both on a short term tactical level, i.e. on specific projects where there is no established usability framework, and in the longer term, where there is the opportunity to shape the contracts signed with offshore suppliers.

There will always be a need for test management expertise based in client countries, working with the offshore teams, but it will not be the same job we knew in the 90s.

References

[1] NASSCOM Industry Trends, IT-BPO Sector-Overview. http://www.nasscom.org/Nasscom/templates/NormalPage.aspx?id=54612 [accessed 9th February 2009]

[2] Nielsen, J. "*Offshore Usability*". http://www.useit.com/alertbox/20020916.html [accessed 9th February 2009]

[3] Lewis, C., Rieman, J. "*Task-Centered User Interface Design: A Practical Introduction*". University of Colorado internet book, 1994. www.hcibib.org/tcuid/tcuid.pdf [accessed 9th February 2009]

[4] Artman, H. "*Procurer Usability Requirements: Negotiations in Contract Development*". Proceedings of the second Nordic conference on Human-Computer Interaction (NordiCHI) 2002. http://www.nada.kth.se/kurser/kth/2D1622/03_04/ArtmanNORDIC.pdf [accessed 9th February 2009]

[5] Holmlid, S. Artman, H. "*A Tentative Model for Procuring Usable Systems*". 10th International Conference on Human-Computer Interaction, 2003. www.nada.kth.se/~artman/Articles/Conference/HCIint03.pdf [accessed 9th February 2009]

[6] Grudin, J. (1991). "*Interactive Systems: Bridging the Gaps Between Developers and Users*". Computer April 1991 (Vol. 24, No. 4) doi.ieeecomputersociety.org/10.1109/2.76263 (subscription required) [accessed 9th February 2009]

[7] Grudin, J. (1996). "*The Organizational Contexts of Development and Use*". ACM Computing Surveys. Vol 28, issue 1 (Mar. 1996), pp 169-171. doi.acm.org/10.1145/234313.234384 (subscription required) [accessed 9th February 2009]

[8] Iyengar, J. "*Usability Issues in Offshore Development: an Indian Perspective*". Usability Professionals Association Conference, 2007. http://www.upassoc.org/conference/2007overview.html (UPA membership required) [accessed 9th February 2009]

[9] Henry, P. "*Advancing UCD While Facing Challenges Working from Offshore*", ACM Interactions, March/April 2003. http://portal.acm.org/citation.cfm?id=637848.637861 (subscription required) [accessed 9th February 2009]

[10] Cronin D, "*Designing for Offshore Development*", Cooper website, 2004. http://www.cooper.com/journal/2004/06/designing_for_offshore_develop.html [accessed 9th February 2009]

[11] Nielsen, J. "*Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier*", 1993. http://www.useit.com/papers/guerrilla_hci.html [accessed 9th February 2009]

[12] Krug, S. (2006). "*Don't Make Me Think!: A Common Sense Approach to Web Usability*". 2nd edition. New Riders. Also, http://www.sensible.com/workshops.html [accessed 9th February 2009]

[13] Constantine, L & Lockwood, L. "*Software for Use*". Addison Wesley 1999.

[14] Lockwood, L. "*Collaborative Usability Inspecting - more usable software and sites*", 1999. http://www.foruse.com/presentations/inspections.pdf [accessed 9th February 2009]

[15] Constantine, L. "*Usability in a Post-Waterfall World*". International Workshop on the Interplay between Usability Evaluation and Software Development, Pisa, September 2008. http://www.labuse.org/files/keynotes2008/IUSED2008keynoteDist.pdf [accessed 9th February 2009]

[16] Cooper, A. (2004). "*The Inmates Are Running the Asylum: Why High-tech Products Drive Us Crazy and How to Restore the Sanity*". Que 2004.

James Christie

9th February 2009

Biography

James lives in Perth in Scotland . He is currently working as a consultant through his own company, Claro Testing Ltd (www.clarotesting.com).
James has 24 years commercial IT experience, mainly in financial services, with the General Accident insurance company and IBM (working with a range of blue-chip clients) throughout the UK and also in Finland.
His experience covers test management (the full life-cycle from setting the strategy, writing the test plans, supervising execution, through to implementation), information security management, project management, IT audit and systems analysis.
In 2007 he successfully completed an MSc in IT. His specialism was “Integrating usability testing with formal software testing models”. He is particularly interested in the improvement of testing processes and how the quality of applications can be improved by incorporating usability engineering and testing techniques.
James is a Chartered IT Professional and a professional member of the British Computer Society (MBCS). He holds the ISEB Practitioner Certificate in Software Testing and is a member of the Usability Professionals Association.

The impact of the global financial crisis on offshore outsourcing

by Nadica Hrgarek

Offshore outsourcing of software development is a growing business worldwide. Increased competition, pricing pressures, R&D not cost effective, more sophisticated customers, not finding the right talent locally or at high cost, faster time to market, are just a few of many challenges facing companies today. In addition, many major corporations are affected by the ongoing global financial crisis and are looking for new efficient ways of cutting costs and improving customer satisfaction. Should companies use the benefits of offshore partnerships to address these challenges?

Introduction

As a result of the recent global financial crisis and rising costs, companies today are forced to react fast to the changes in the market trend. They need to be flexible and open to new cost-effective approaches to win the competitive game on the market and to improve the overall quality of products/services. Companies need to look in new locations with lower labor costs to reduce overall costs. In addition, companies who want to compete globally must comply with regulatory requirements in the countries in which their products/services are sold. IT companies seeking international markets and needing to localize their products to specific platforms and languages very often use offshore development resources in or near their target markets.

Outsourcing involves the transfer of the responsibility for carrying out an activity (previously carried on internally) to an external service provider regardless of the provider's location. Generally speaking, the main goals of outsourcing are: cutting operational costs and reducing risks, availability of resources, using advantages of new technologies, using competence centers, better IT service, more effective way of using existing resources through focus on core business, process improvement, etc.

Offshore outsourcing (offshoring) refers to the international relocation of jobs and processes from domestic to overseas locations. According to Wikipedia, offshore outsourcing is the practice of hiring an external organization to perform some business functions in a country other than the one where the products or services are actually developed or manufactured.

Many companies use offshore outsourcing in the most cost-effective location to reduce operating costs, to improve quality and to reduce time to market. The top offshore development centers providing offshore services are located in countries around the world: India, Russia, China, Pakistan, Ireland, Singapore, Vietnam, Philippines, Sri Lanka, New Zealand, Malaysia, Brazil, Ukraine, Romania, Slovakia, Hungary, Bulgaria, Poland, the Czech Republic, etc.

The A.T. Kearney Global Services Location Index (GSLI) compares 50 global offshore locations against more than 40 metrics across three major categories: financial attractiveness (40%), people and skills availability (30%), and business environment (30%). The top ten offshore locations in 2007 according to new A.T. Kearney study [1] are: India, China, Malaysia, Thailand, Brazil, Indonesia, Chile, Philippines, Bulgaria and Mexico. Table 1 provides a list of Gartner's 30 leading locations for offshore services in 2008 by region.

Challenges in offshore projects

Offshore outsourcing usually takes place with distant countries and may involve a significant time change, cultural differences and other difficulties in collaboration. Offshore software development projects can fail due to the same reasons as any other IT project. However, the risk that a project may fail is higher and the companies shall anticipate what they must do to minimize these risks.

North America and Latin America	Asia/Pacific	Europe, the Middle East and Africa
Argentina	Australia	Czech Republic
Brazil	China	Egypt
Canada	India	Hungary
Chile	Malaysia	Ireland
Costa Rica	New Zealand	Israel
Mexico	Pakistan	Morocco
Canada	Philippines	Poland
	Singapore	Romania
	Thailand	Russia
	Vietnam	Slovakia
		South Africa
		Spain
		Ukraine

Table 1: Gartner's 30 leading locations for offshore services in 2008 by region

Gartner five reasons why offshore projects fail
• Unrealized cost savings
• Loss of productivity
• Poor commitment and communications
• Cultural differences
• Lack of offshore expertise and readiness

Table 2. High risks in offshore projects according to Gartner [3]

According to the CHAOS research report [12] published in 1994, only 16,2% of IT projects in the US were successful, while 52,7% were challenged (completed and operational but over budget, over the time estimate and offers fewer features than originally specified) and 31,1% were impaired (cancelled at some point during the development cycle). The CHAOS report points several factors that cause projects to be challenged: lack of user input, incomplete requirements and specifications, changing requirements, lack of resources, unrealistic expectations, etc. However, more than 50% of the major factors presented in the CHAOS report that cause software projects to fail are related to requirements. Incomplete requirements and lack of user involvement are ranked at the top of the list. In almost all IT projects, requirements are subject to change. The 2000 updated CHAOS research report [11], entitled "Extreme Chaos", found 23% projects failed, 28 % succeeded and 49% challenged.

The challenges in offshore development projects can range from language barriers, poor project management, time zone differences, immature or non-standard software development processes to cultural differences. Table 3 provides an overview of typical challenges and key factors to ensure success of offshore software development projects.

Use of IT outsourcing

Demand for skilled IT professionals continues to grow, while education systems in many advanced countries are producing an insufficient quantity of educated IT professionals. The growing domestic shortage of qualified IT professionals is becoming the most important reason for offshore software development. Other key drivers for companies offshoring software development are increasing time to market and labor cost savings. A few years ago, Forrester Research [8] recommended using Agile processes for offshore software development.

There are many different kinds of outsourcing. For example, if you search for software outsourcing, Google will approximately return 11 million pages. According to location we can distinguish the following IT sourcing models: *onsite sourcing, onshore/domestic sourcing (onshoring), nearshore outsourcing (nearshoring), offshore outsourcing and global sourcing*. If we consider the chronological aspect, the following IT sourcing models are possible: *insourcing, outsourcing and backsourcing*. Basic types of IT sourcing models in regards to the business orientation level are: *IT outsourcing* (e.g. maintenance, consulting, databases, infrastructure, etc), *business process outsourcing* (BPO) (e.g. call centers, marketing, finance, accounting, payroll, HRM services, training, supply management, data entry, back office operation, etc), *offshore software development* (OSD) and *knowledge process outsourcing* (KPO). To create the most value from outsourcing, the right question is how to choose and apply IT sourcing model which best meets the company's business needs.

Forrester Research [7] predicted in 2004 the following areas as the future in offshore IT outsourcing: enterprise packaged applications (e.g. implementation of SAP and other ERP systems), IT infrastructure (e.g. DBA services, database tuning, network monitoring and management, system management, system software support, etc), help desk and data warehousing systems.

Key IT outsourcing areas for effective use of outsourcing resources may include, but are not limited to:

- software development,
- web design, web application development and maintenance,
- IT security management,
- testing and quality assurance services,
- test automation,
- validation services for life science industry,
- graphics design,
- web hosting,
- server hosting,
- network backbone services,
- IT remote support (e.g. network monitoring and support, remote monitoring),
- product support (hardware, software),
- help desk,
- consulting services (e.g. SAP consulting services, validation consulting services, training),
- project management,
- database management.
- document/content management,
- internet marketing services,
- e-commerce projects,
- business process reengineering (BPR) services, etc.

Challenges	Key success factors
• Cultural challenges (e.g. stronger hierarchical organization in Asian countries) • Difficulties in sharing software assets • Distributed teams (difficult face-to-face communication) • Dynamic business requirements and priorities • Focus on technology and not on business needs • Hidden costs associated with transitioning and offshore management (e.g. travel, staff turnover, communication, efficiency, process changes, etc) • High employee turnover rate in offshore service center • Lack of customer participation • Language • Poor knowledge transfer from onshore to offshore team • Poor quality management • Poor project management • Recruiting the right offshore personnel • Time zone differences • Legal and intellectual property issues • Trust (e.g. handling of confidential company and customer data)	• Effective project management and avoiding project scope creep • Effective training of the offshore team • Establishing shared development environment • Having adequate user/system specifications and detailed project plan • One transparent risk management process for onshore and offshore team • Political and economic stability • Regularly scheduled and interactive customer collaboration via multiple communication channels (e.g. video and conference calls, chat sessions, web conferencing, Skype, e-mail, phone, Wiki, application sharing tools, document repository) • Relationship building trust • Rotating team members on all levels, not only executives • Setting up distributed knowledge management • Software development in iterations with short delivery cycles • Starting small and testing the chosen service provider with a trial project • Use of quality standards (e.g. CMM/CMMI, ISO 9001, ISO 20000, ISO 27001, etc)

Table 3: Challenges vs. key success factors of OSD projects

Outsourcing of validation activities

Manual software validation tests in the regulated industry (e.g. medical device, pharmaceutical and biotech industry) conducted by the in-house validation engineers or external consultants can double a company's compliance costs.

Manual testing is slow, intensive and often causes inaccurate test results because it is being done by operators. If the company is required to perform software validation of many systems in a short period of time, outsourcing of validation consultants can help to automate software validation process by implementing automated tools for validation and validation management.

If the company doesn't have trained software validation specialists, it is recommended to involve system validation consultants who can provide validation expertise and experience with all types of computerized systems across the organization. Their job is to help ensure regulatory compliance with software validation requirements and reduce overall project costs. System validation consultants can help developing software validation policies and procedures, executing software tests and documenting the test results (e.g. test plan, test protocols, test scripts, test reports). They may also provide the following services: training on computer system validation, conducting software vendor audits, computer system implementation support, legacy system validation, ensuring FDA 21 CFR Part 11 compliance, etc.

Impact of the global financial crisis on offshoring

There are many different opinions about the impact of the global economic crisis on offshoring. Key questions are how to face the global financial crisis and transfer the crisis into opportunity.

According to McKinsey's survey [10], the global offshoring and outsourcing industry, which relies on demand from the financial services industries in the U.S. and EU, has been hit hard by the global financial crisis. As a result, the value and number of outsourcing contracts in the financial industry has been reduced. McKinsey reports [10] that of six major U.S. financial institutions recently interviewed, five plan to maintain or step-up their IT offshoring and outsourcing plans over the next 6 to 12 months.

Outsourcing countries like **India** which deliver software products/services with the high percent of the exports to the U.S. shall consider that investment into IT can decline in the next 12 months. The United States and Canada are India's largest export markets for computer software and services. Indian exports are mostly in the form of providing offshore software services. According to A.T. Kearney study [1], India is the leading destination for offshore services. However, Gartner research predicts that by 2012, India's dominant position as an offshore location will be significantly diluted by effective alternative locations.

In 2004 Forrester Research [9] has concluded that 1.2 million European jobs will move offshore by 2015. Due to the global financial crisis these numbers may even be higher than forecasted. According to Forrester Research [9], IT workers will take the biggest hit (approximately 150.000 jobs). There are many benefits of moving jobs to overseas locations. For example the average annual salary for a system architect in the UK was €130.000 in 2004, compared with a just €40.000 average salary for a comparable system architect in India. Another example by Gartner [3] shows that an application maintenance worker in India earns about $25 per hour, compared with $87 per hour in the U.S.

In 2007 Gartner research has predicted that by 2010, 30 percent of Fortune 500 enterprises will source from three or more countries.

According to A.T. Kearney study [1], **China** is ranked 2nd in list of top 50 global offshore locations. Many Chinese offshore IT companies have achieved CMM/CMMI level 4 or 5, ISO 9001 and other certifications to demonstrate their strength and high commitment to quality. New research survey from McKinsey [10] shows that between 2004 and 2007, the number of Chinese companies with CMM/CMMI level 4 or 5 certification grew by 39% per year. Also the number of companies passing ISO 20000 (ITIL) certification has been growing in recent two years. According to survey of 75 leading software and IT companies in 14 cities across China by McKinsey [10], the biggest challenge for Chinese service providers is lack of experienced talent familiar with the offshore business in China. McKinsey reports that the financial crisis will create a unique opportunity for Chinese companies to fill capability gaps with highly qualified talent.

Positive macroeconomic trends in **Russia** are opening opportunities in a variety of sectors including the IT (i.e. software, hardware and services) and telecommunication sector. Russia is often mentioned as a rapidly emerging offshore country for software development. According to A.T. Kearney study [1], Russia is ranked 37th out of 50 locations worldwide. Russian salaries are significantly lower than those in Western Europe and the U.S. Russia has an excellent education system and a large pool of skilled technical specialists and scientists with mathematics, physics, engineering, and computer science backgrounds. They are capable to solve complex and math-intensive problems. Russian software developers are well suited for design and development of complex, innovative, high risk or critical real-time software systems (e.g. aerospace, military applications, automotive, healthcare and medical devices, precision electronics, security, online banking, etc).

CEO of Russian information security software vendor Kaspersky Lab and leading antivirus expert, Eugene Kaspersky, expressed his concerns at a press event in December last year that some software engineers who lose their jobs due to the credit crunch will turn to cybercrime. He pointed the rising trend of more sophisticated Trojan viruses and other malware attacks being propagated through the Internet to steal personal data, to get access to bank accounts and use stolen data for fraud transactions. According to Khalid Kark [4], a principal analyst at Forrester Research, the average security breach can cost a company between $90 and $305 per lost record. The total costs to the company may run into millions of dollars.

A recent study report [13] published by the Forrester Research, an independent technology and market research company, notes significant trend and continued evolution of cybercrime on the Internet (e.g. malware, spam, phishing, emerging threats). Gartner research study [5] points 75% of hacks happen at the application level. Many web 2.0 applications / web services are vulnerable and therefore an easy target for attacks. Some examples of the most dangerous attacks at the application level [6] are: cross site scripting (XSS), SQL injection, improper authorization, hard-coded password, etc.

Summary and conclusion

Due to the global financial crisis, economic indicators point to moderately slower global growth. The effects of the economic crisis can be felt in the offshoring and outsourcing industry too, but the crisis should be transferred into opportunity. According to a study reported by Corbett [2] the reduction of costs (36%) and focus on core competencies (36%) are the most significant reasons for outsourcing cited by large U.S. companies. As companies try to reduce costs due to the ongoing economic crisis, demand for outsourced/offshored services is expected to increase in 2009.

Some of the main factors when selecting the most appropriate offshore company may include, but are not limited to: quality and skills of the IT professionals, low labor costs, high level of expertise and domain knowledge, experience, geographical proximity, cultural affinities with the western countries, low attrition rate, very good foreign language skills of the local people, ease of travel, well developed IT support infrastructures, data security (e.g. use of firewalls, data control, data encryption), security of intellectual property, favorable government regulatory environment supporting offshore development activities, quality standards, etc.

Offshore outsourcing is a good way to go for organizations needing software product/services if the company providing offshore services has good references and is reliable. Offshore software development is not ideal solution for development of complex (e.g. ERP), security and high technology risk software systems in the regulated industries (e.g. medical device, aviation, military, biomedicine, etc). When using offshore software development, the onshore development process should be established and in place (e.g. CMMI level 3 and higher).

References

[1] A.T. Kearney, Inc. Offshoring for Long-Term Advantage: The 2007 A.T. Kearney Global Services Location IndexTM. Report number ATK807022, 2007

[2] Corbett, M. F. & Associates: Taking the Pulse out of Outsourcing – Data and Analysis from 2001 Outsourcing World Summit. December, 2001

[3] Gartner, Inc.: Gartner: Five reasons why offshore deals go bust. 2005 <http://www.computerworld.com/managementtopics/outsourcing/story/0,10801,102677,00.html>

[4] Kark, K.: Calculating the Cost Of A Security Breach. Forrester Research, Inc., April, 2007

[5] Lanowitz, T.: Now Is the Time for Security at the Application Level. Gartner, Inc., December, 2005

[6] Martin, B. et al.: 2009 CWE/SANS Top 25 Most Dangerous Programming Errors. The MITRE Corporation, January, 2009

[7] McCarthy, J.C. et al.: Offshore Outsourcing: The Complete Guide. Forrester Research, Inc., September, 2004

[8] Moore, S., Barnett, L.: Offshore Outsourcing And Agile Development. Forrester Research, Inc., September, 2004

[9] Parker, A.: Two-Speed Europe: Why 1 Million Jobs Will Move Offshore. Forrester Research, Inc., August, 2004

[10] Peng, A., Benni, E., Hu, A.: Destination: China – A Perspective on the Offshoring and Outsourcing Industry. McKinsey & Company, January, 2009

[11] The Standish Group: Extreme CHAOS. The Standish Group International Inc., February, 2001

[12] The Standish Group: The CHAOS Report. The Standish Group International Inc., December, 1994

[13] Wang C., Penn J, Dill A.: Threat Report: The Trends And Changing Landscape Of Malware And Internet Threats. Forrester Research, Inc., June, 2008

Index Of Advertisers

Cognizant	18
Díaz & Hilterscheid GmbH	32, 33, 47, 74, 93, 102, 104
eXept	63
FRHACK	24
ImproveQS	39
iSQI	10
ISTQB	52-53
Kanzlei Hilterscheid	101
Microsoft	56
PureTesting	102
RCBS	36
SELA	97, 102
Testing Experience	27, 31, 37, 41, 59, 76, 91
Test Planet	102

Biography

Nadica Hrgarek holds a B.Sc. in information systems and a Master of Science in information science from the University of Zagreb, Croatia.
Since March 2007 she is a member of the Quality Assurance department at MED-EL Elektromedizinische Geräte (www.medel.com), an Austrian medical device company located in Innsbruck. Nadica is currently working as a quality improvement specialist. Her responsibility covers all aspects of quality improvements ranging from coordination of investigations, corrective and preventive actions, non-product software validation support, conducting training, internal and supplier audits, etc.
Before joining MED-EL she was a software quality engineer at Fabasoft R&D Software (Linz), a system analyst and quality specialist at InfoDom (Zagreb) and an organization assistant at Mercator-H (Velika Gorica).
She holds the ISO 9000 internal and lead auditor, ISTQB Certified Tester Full Advanced Level, IREB Certified Professional for Requirements Engineering and iSQI Certified Professional for Project Management Foundation Level certificates.
She is co-founder and advisory board member of the Croatian Testing Board (CTB) founded in 2008. She is also a member of the German Association for Software Quality and Training (ASQF).

Interview Dr Mike Bartley

Dr Mike Bartley,
PhD in Mathematics at Bristol University and
José Díaz, Editor

1. When did your firm decide to go for an outsourcing vendor?

In May 2006, ahead of my recruitment. It was decided to recruit a Manager with specific responsibility for software testing (and pre-Silicon hardware verification) and to ask that Manager to have as much of the team offshore as possible.

2. What were the business criteria that were considered before coming up with the decision?

There were many, but the main ones were to improve company focus; to improve company resource flexibility and in doing so, reduce time to market; software quality improvement; and finally cost variablization and cost reduction.

3. What were the IT processes you planned to outsource and why?

We were looking to outsource our manual software testing. We had built up an extensive internal automated build and test process suite which allowed us to perform daily build and test. But there were a number of tests that were either too hard or not cost-effective to automate.

4. How did you go about the vendor selection process, and what were the basic criteria for evaluating the vendors?

Well, that is a long story which I cover in my paper elsewhere in the magazine. We were not just looking for a transaction-based outsourcing relationship, we were looking to enhance our software testing and bring real business level benefits. So we looked for "partners" rather than "vendors", people that we could build a long-term relationship with. But as I said, you need to read my paper to get the details.

5. What were the key expectations from the outsourcing activity?

We mainly wanted to improve time-to-market and software quality – things that would have a real impact on the business. It annoys me when people view software testing purely as a cost centre. Testing can add real value to a business when it is done well – so aim high!

6. Who were the vendors short-listed (nature of vendors/ size)?

There was a lot of variety in the short-list. It included large companies (over 10,000 employees) and small (those with less than 100 staff); specialist testing companies and large general-purpose IT service providers.

7. Why were Development and Testing split? What were the reasons for going for independent testing? How was it beneficial?

When I first arrived, they weren't split. The main reasons for splitting them were to free up the internal developer team (who were doing the testing) and to improve quality. There is also very strong evidence that separating the development and testing along management, technical and financial boundaries is a major contributor to software quality improvement.

8. Would you recommend that others outsource their software development and/or testing?

I now spend my time helping companies to do exactly that. The potential business benefits are huge and can be had by both large and small, young and old companies. But the risks are high, too. My paper in the magazine describes how to maximize your chances of success.

MBT as the next step in testing!

by Elise Greveraars

Model Based Testing (MBT) has become a very popular term these last few years. It is best known in the world of embedded systems. But what exactly is MBT, what is the added value of MBT for the administrative world and what are the prerequisites to start using MBT in your organization?

First let's have a look at the definition of MBT. MBT can be defined as following:

Model based testing is software testing in which a functional test model is created that describes describe some of the expected behavior (usually functional) of the system under test (SUT). The purpose of creating these test models is to review the requirements thoroughly and/or to derive test cases in whole or in part from the test model. Generated test cases can be used for manual and/or automated test execution. The test models are derived from the requirements.

Within MBT a distinction can be made between informal and formal MBT. The difference between formal and informal MBT is that formal MBT uses formal test models that comply to certain standard modeling rules while informal MBT doesn't use formal test models. Test cases can be automatically generated from formal test models and must be manually generated from informal test models.

Informal model based testing

Often testers start drawing some sort of test model while reading requirements. These test models help to get a better understanding of the requirements which are often only described in textual documents.

While drawing these test models many questions will arise. Sometimes because specifications contradict each other, contain incorrect information, are unclear - or even worse - not present. By drawing test models and asking questions, the tester will gain a good insight into the system to be tested and the tester will help the functional designer to get the specifications right.

Drawing such test models will help improve the quality of the requirements and will assist in finding defects at a very early stage in the process. Creating test models is also one of the first process steps in informal model based testing. The grey box in Figure 1 indicates the scope of informal MBT.

So what exactly are the process steps of informal MBT?

Before starting to create test models it is important to consider what functionality needs to be modeled and in which order. The answer to this consideration can be found in the Risk Analysis (RA). The RA should be the basis for each test project. It contains information about which specifications have a high risk score for a certain version of the system. Based on this information a choice can be made on what to model, for example only model the highest risks requirements or model the highest risks requirements first and the lower risks requirements later. The creation of a RA is preferably done in the business analysis stage but is possibly performed later.

Once the risk information is clear the modeling can start. The notation of test models used for informal model based testing doesn't have to comply with certain modeling rules. The only

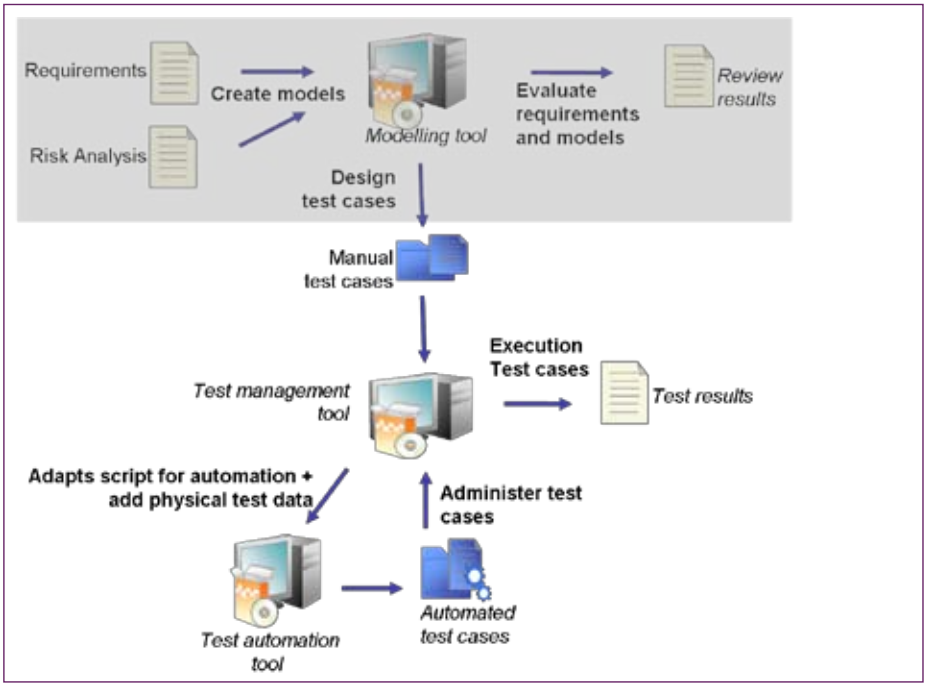


Figure 1: Process steps informal model based testing

rule for modeling is that the test models to be created are readable for the parties/persons involved in testing. Parties or persons involved can be other testers, functional designers, end users etc. These parties will have to determine themselves what is a readable form. Any tool able to create test models can be used for informal model based testing.

models and transform them into test cases.

An example of a formal modeling notation is UML but many other modeling notations exist.

Figure 2 shows the process steps for formal MBT. As with informal MBT it is important to consider what functionality needs to be modeled and in which order. This information

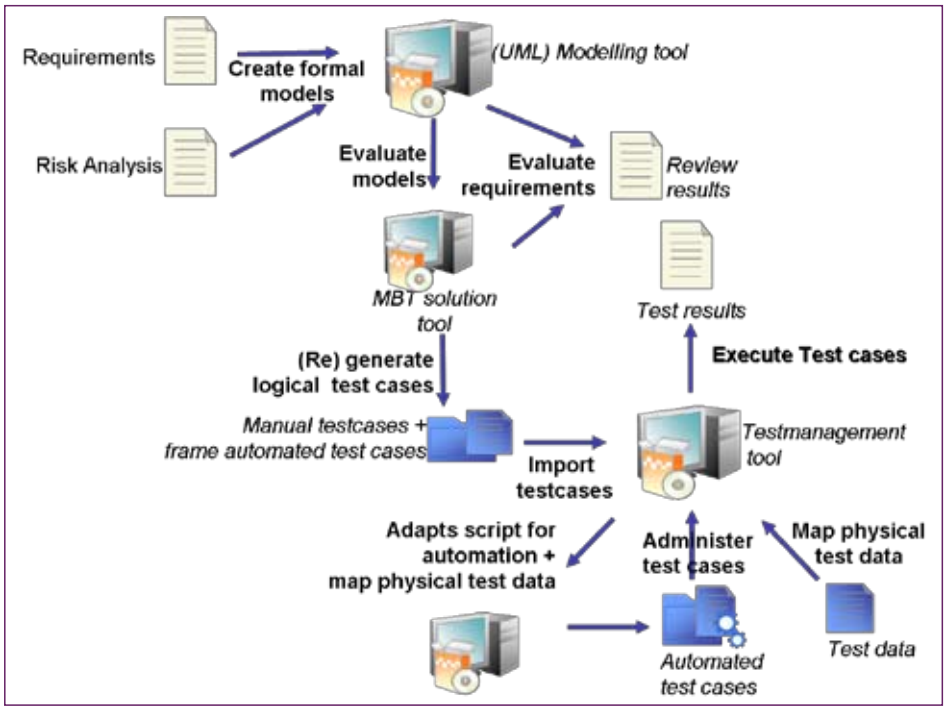


Figure 2: Process steps formal model based testing

During the creation of those early test models, incorrect, unclear or missing specifications will be found. This information should be registered and communicated to the parties involved. When needed the specifications should be updated and released. After each release of new specifications the test models need to be screened and adjusted to reflect the changes that have been made. This can be an iterative process.

Once the specifications and the test models are approved, the design of the test cases can commence. The test models will be used as a guideline for designing the test cases and will give the test designers a good overview of test cases to be designed. The design of test cases is a manual action when using informal MBT.

Test cases will generally be stored in a test management tool to be executed manually or they will be adapted for automation test cases to be executed automatically.

Formal model based testing

One of the biggest advantages of formal MBT is the possibility to automatically generate test cases based on test models. A MBT tool is needed to generate those test cases automatically.

Formal MBT uses strict notation guidelines for the models. This ensures compatibility with the test generation tool, which can check and interpret the test

should be extracted from the Risk Analysis (RA). Based on this information a choice can be made for example to only model the highest risks requirements or to model the highest risks requirements first and the lower risks requirements later.

The test models to be designed must be formal and accurate. To be able to design such test models more sophisticated modeling tools are needed, for example Borland Together or IBM Rational Software Modeler. Sometimes model based testing tools also offer modeling functionality. In this article the assumption is made that a separate tool is used for modeling purposes.

During the modeling phase incorrect, unclear or missing specifications will be found. These specification defects need to be registered and communicated to the parties involved. In formal MBT the test models will also be offered to the model based testing tool for simulation. By simulating the test model the specifications will also be evaluated. Simulation also helps in finding interpretation defects which are possibly introduced by designing the test model. The findings of the simulation should also be registered and communicated to the parties involved.

The specifications and test models will have to be fine-tuned based on the findings and will have to be approved by parties/persons involved in testing and capable of understanding the formal test models. Parties or persons involved can be for example functional designers.

Once the test models are designed and approved the model based testing tool can generate the logical test cases.

The generated test cases can be for example HTML or XML output. Some model based testing tools also offer a plug-in for a test management/execution tool to make it possible to import test cases into the test management tool. The generated test cases will be logical test cases to be used for manual execution or a framework with automated test cases.

The generated logical test cases can be adapted to physical test cases by mapping test data after which the test cases can be executed manually.

To automatically execute test cases the generated framework of automated test cases should be adapted. The generated test cases are abstract test cases and should be made concrete to make them executable. Abstract field names should be mapped to real field names used in the system, test data should be mapped to test cases etc. The adapted test cases will be administered in the test management tool and can be executed automatically.

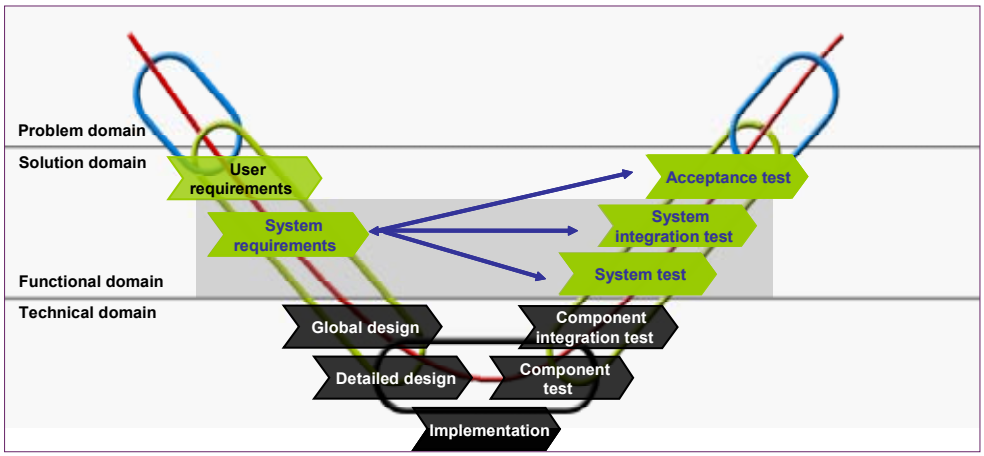


Figure 3: Position of formal MBT in V-model

Positioning of informal and formal MBT in the V-model

Now we have an idea about the different forms of MBT, let us find out the position of MBT in the V-model, see Figure 3.

In the problem domain of the V-model the wishes, opportunities, problems and policies are identified. These will be converted to a solution in the solution domain. The solution domain can be split up in a functional and technical domain.

Informal MBT can be used for both the functional domain and the technical domain within the solution domain. It will probably be used most in the functional domain for system, system integration and acceptance testing.

Test models as created in formal MBT are mostly based on the system requirements, see the grey box in Figure 3. This makes formal MBT most suitable for system and integration testing. Formal MBT is less suitable for acceptance testing since acceptance testing should be based on user requirements. However, parts of the test cases generated by MBT can possibly be re-used for acceptance testing.

The most common kind of testing with MBT in general is functional testing but can also be some types of robustness testing. Automated test cases generated from formal models can possibly also be used for performance testing. MBT is less useful for usability testing.

Test organization

When applying MBT it is very important to have the commitment of all parties, this includes management. MBT requires an adapted way of working from the traditional testing. Test models will have to be designed in an earlier phase of the project compared to the traditional design of test cases. Much interaction will be needed between the functional designer and the tester. Management will have to support and propagate this way of working. They will also have to invest in training and possibly in tooling; commitment is essential.

Model based testing also requires additional skills to traditional testing skills. For this reason a new role or function is introduced for MBT, the test constructor.

The definition of a test constructor is:

A test constructor is a person who builds or forms the necessary test models by following precise specifications, using a good level of abstraction for test purpose and using tools and techniques for modeling and the generation of test cases.

Required skills for the test constructor are modeling skills, skills on transforming written specifications into models and a basic knowledge regarding programming. The test constructor should also have domain and testing knowledge and should have good social skills. He or she will have much interaction with the different parties involved in the project to cross check the models with the available information.

Tools

There are already many commercial and non-commercial MBT tools on the market. The purpose of these tools is to generate test cases based on test models. Some tools also support modeling functionality and/or test execution functionality.

The test tools can be grouped in online and offline model based testing tools. Offline model based testing tools will generate a finite set of tests to be executed later. This allows automatic test execution in third party test execution platform. Online model based testing tools will connect directly to a system under test and test it dynamically. Online model based testing tools are common in the embedded industry while offline model based testing tools are more common in the administrative domain.

The different tools available support different models notations like, for example, UML, OCL, B notation, Spec#, FSM, etc. Some tools also offer plug-ins for modeling tools like for example Borland Together or IBM Rational Software Modeler. This plug-in will check the consistency of models.

Each tool also provides its own test case output. Test cases are generated as HTML, XML, ASCII, Comma Separated, TTCN-3 output etc. Some tools also offer plug-ins like for example exporting test cases to HP Quality Center, etc.

Case studies

Informal model based testing for review purposes

In this case study a comparison is made of specification defects found in two versions of a newly built application. Both versions were designed by the same functional designer and had been previously informally reviewed.

Based on function points both versions are of an equal size. For version 2 informal test models were created as part of the activity intake test basis, for version 1 no test models were created.

The results of this case study show that in version one, out of all defects 11% were specification defects found prior to testing, see Figure 4. In version two, out of all defects 26% were specification defects found prior to testing, see Figure 5.

When evaluating the types of specification defects in version 2, the following types of specification defect were found:

- Functional defects
- Defects about unclear specifications
- Textual defects

The production severity of the specification defects found ranged from blocking to cosmetic.

At the end of the case study we found that a few specification defects were not found when creating the test models for version 2. The missed specification defects were related to screen specifications which were not well defined. Because it is not possible to model the screen specification, it was not surprising that these specification defects were missed.

Lessons learned

One of the lessons learned during this case study was that it is very useful to create test models to support the review process. A lot of missing and unclear specifications were identified early. Some time had to be invested for the creation of the test models but this time paid dividends in the later phases due to the knowledge the tester had gained early on by creating the test models. This knowledge was very useful to the testers during the design and execution of test cases.

Another lesson learned was that it is important to start modeling during the review process and not as part of the activity "intake test basis" as was done in this case study. At this stage the

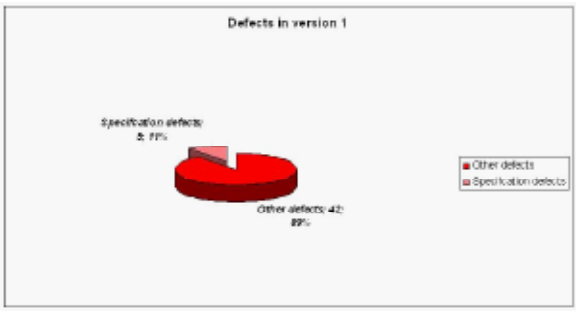


Figure 4: Specification defects in version 1

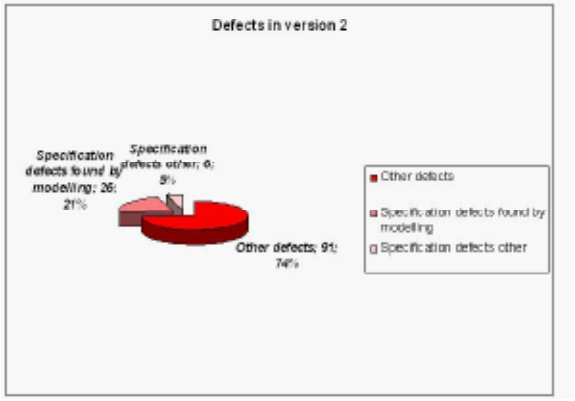


Figure 5. Specification defects in version 12

functional designers do not want to review the created test models anymore. They consider this as double work as the specifications have been finalized. When starting to model during the review sessions more commitment can be expected from the functional designers on reviewing the test models.

Formal model based testing

The purpose of this case study was to gain knowledge on formal MBT and to judge the usability of formal MBT. The case study was performed in cooperation with the model based testing tool supplier Smartesting.

For this case study a team of test specialists was formed and trained in 3 weeks on formal modeling and tools to be used. The approach used was incremental and iterative, starting small and simple and expanding gradually. By applying this approach the team could quickly get an impression of using MBT and gradually gain knowledge on the test models to be designed and tooling to be used. At the same time another test team was working in parallel on the same project using traditional test design.

Lessons learned

The case study was very successful in gaining knowledge of MBT. The testers were very enthusiastic to learn about MBT. For them MBT was a new and exiting challenge. They found modeling to be a very creative task which could be learned easily when education in modeling and tools is provided. In order to design the test models correctly much interaction was needed with other parties.

The test models gave the testers a better overview than the textual specification and were very useful in communicating with functional designers. However, knowledge is needed to understand the test models. For this reason the test models used are not considered to be business user friendly.

The test generation did save a lot of time on designing test cases manually. The traditional test team created 38 test cases in 70 hours while the MBT test team created 47 test cases in 24 hours. On top of this the MBT team found additional findings in the requirements compared to the traditional test team.

The greatest added value of MBT is to be gained when using test automation. Beside manual test cases Test Designer of Smartesting will automatically generate a framework of automated test cases which will reduce the time to create/design automatic test cases significantly.

At the start of the case study time was required to design the test models. The initial set-up of the test model was more time consuming than updating the test models.

Unfortunately it is not possible yet to make a difference in the test generation for high or low risks requirements. Currently test cases are generated from the requirements as if they all have the same risk level. When executing test cases manually, this means that the test cases have to be sorted out by hand. The only option to reduce the number of generated test case is to reduce the modeling of requirements.

The tool Test Designer from Smartesting was found easy to use. The tool supports the usage of references to requirements in the test models. These requirements with linked test cases generated from the test model are administered in HP Quality Center. This makes traceability possible between the test models, requirements, test cases and test execution.

Conclusions

By designing test models knowledge is gained on the system and defects are found at a very early stage. This will reduce much development and testing time. It is important to model risk based, so start modeling the high risk requirements.

The test models are very useful for the creation of test cases. This is the case for formal and for informal test models. Formal models are practically useful since these can be used for automated test case generation. This will heavily reduce the maintenance on test cases; this is especially the case when applying automated testing. Instead maintenance on test models will be needed. Maintaining test models will be by far less time consuming than maintaining large test repositories.

To be able to construct formal models a model based test designer, the test constructor, will be needed in addition to the traditional roles or functions. A test constructor requires additional skills to traditional testing skills.

Formal MBT will provide very high test coverage but it is not possible yet to generate test cases risk based. Model based testing tool providers will probably implement risk based test generation in the near future.

In general MBT can certainly help to improve the test process and test results especially on the level of system testing and system integration testing. Reduction in time and costs are to be expected once people are trained in modeling and the initial test models are designed. Because of this the adoption of formal MBT will continue to increase.

Recommendations for starting to use MBT

So now you want to start using MBT, how to best start?

Informal MBT can always be used. Tooling to design test models for example will be handy but are not required for informal MBT. First try to find out what are the most important requirements. Once these are clear start reading the specifications and just start creating test models. Be surprised by the questions that will arise and the knowledge that will be gained by doing so. Once these test models exist, discover how easy it is to use them as the basis for designing the test cases.

A first step can also be to start a small project to gain experience on formal MBT. This will require the necessary tools like a modeling tool, an MBT tool for the generation of test cases and test management and execution tooling. Decide what model notation to use and make sure the people who will work on this project will get a proper training in modeling, MBT and tooling.

Start modeling as early as possible in the project and start to model simple functionality to gain experience. Gradually more complex functionality can be modeled. Test models can get complex when not applying the correct level of abstraction. For this reason it is important to design test models that contain enough test information but will stay relatively small at the same time.

When applying MBT consider if regression tests are to be expected for certain functionality. If no regression is to be expected it might not be useful to invest substantial time in the initial set up of a formal test model. On the other hand take into consideration that investing time in creating the test model will help to find specification defects in an early phase.

And last but not least; just hop on board and get enthusiastic!!

Biography

Elise Greveraars started testing in 1997 by setting up and managing a testing department for an ICT company. In 2004 she started working as a test consultant for Atos Origin. She is part of the national testing competence group and in 2006 she initiated an Atos working group on MBT. She won the “Rising Star Award” @ Eurostar 2008 on her presentation and paper on MBT and is a real “testaholic”.

Interview Sumithra Gomatam

Sumithra Gomatam, Sr. Vice President of Cognizant and
José Díaz, Editor

1. What in your view are the Pros and Cons of Outsourcing?

Outsourcing as a concept has been in existence for quite long; the idea being organizations tend to become more effective and grow fiercely when they focus on their core competences and leave the rest for experts to handle. For example in the case of banks, the core business is banking and financial management, a retail shop is about increasing foot-fall and stock turnover. IT for these organizations is a business enabler, hence a significant amount of investments are made in IT. However, as these organizations grow, their IT organizations tend to transform into cost centers. Overheads get built into the system and, if unaddressed, tend to wipe off margins. Outsourcing of non-core activities to experts will help attainment of quality solutions at optimized cost. However, this demands a lot of synergy to be built into the system. Outsourcing is a strategic initiative, and is about partnering. It involves identification of tasks that can be outsourced, evaluation and selection of vendors, identification of risks involved, mitigation plans, contracts planning and sourcing, continuous evaluation and handling the right portfolio mix. The choice of vendors is the most critical activity as the entire ecosystem comprising of stakeholders will be impacted by the move and performance. Typically organizations set up vendor management offices to handle outsourcing efficiently. Critical parameters such as vendor capability to deliver solutions on time, credibility, reliability, mode of engagement etc are to be evaluated before the decision to engage is opted. In an ideal scenario, a vendor can transition from a mere service provider to a strategic partner. The key driver to success is building a strong relationship resulting in increased trust and confidence. These symbiotic associations have always enabled faster growth of the partnering organizations. However, there are factors that can dent relationships which may be internal or external to the ecosystem. Overlooking these factors can adversely impact the entire move to outsource, reaping little or no return on investment.

2. What levels of market maturity are we in today with respect to IT outsourcing?

To evaluate the market maturity, we need to consider a few critical factors such as velocity of outsourcing by organizations, the number of service providers, their size and capabilities, competition, influencers such as regulatory bodies etc. Typically not all organizations have successfully ventured into IT outsourcing. An industry analyst report claims that less than 20% of organizations have experimented outsourcing, of which a mere 5% have opted for full-fledged outsourcing. This indicates that when competition intensifies, we could witness this percentage increase significantly. Large organizations with global footprints have opted for IT outsourcing in a big way. On the supply side, the market is more mature with large number of players offering a wide range of services. Service providers range from small local players, offshore players and large global players. Service providers find it difficult to build differentiation on the services front, and are embedding differentiators through pricing models and engagement modes. However as the market is still growing, there is no stiff competition witnessed. Every service provider is able to grab their share from the market and serve their clients based on what is needed. We see a trend towards consolidation both on the demand as well as supply front. Service providers today are seen to adopt the merger & acquisition route to build capacity and capability. This again is a clear indicator that the market is actually pacing towards maturity.

3. Why in your view do customers outsource testing?

Software testing involves validation of the application to ensure that the user experience is in keeping with the specifications. Traditionally, companies offered testing services in an integrated fashion, bundling it with application development and application maintenance services. Today, companies like Cognizant offer testing services on a standalone basis, focusing on supporting enterprises' and product companies' testing needs through Independent Verification and Validation (IVV) of their software. In such an outsourced testing model, the testing teams are physically and logically separated and are given full independence to report the test results to the customers. This has been enabled by an effective engagement model where the testing team works closely with the client's Quality Assurance organization. More importantly, the testing team directly reports to business teams and not to the project managers or development heads, thereby giving them the independence and objectivity to deliver value.

Customers tend to outsource testing because of the following reasons:

- This helps augment the capacity of the IT teams at optimized cost owing to the availability of a large talent pool globally.
- This helps manage resource requirements based on need, especially while catering for the sudden rise and fall in demand.
- Industry best practices and defined methodologies help in achieving higher quality.
- The availability of testing tools expertise and infrastructure ensures reduced cost, improved quality and increased speed.
- The metrics captured and reported help the business evaluate the cost of testing.
- Continuous process improvements result in higher productivity and increased cost savings.

- Independent validation helps boost confidence levels during application go-live decision-making.

Outsourced testing models provide the benefits expected in any outsourced IT service model, such as access to a large pool of dedicated professionals with a wide range of skills, and increased efficiencies due to the value creators and solution accelerators developed based on the teams' deep expertise and experience.

4. What are the distinct advantages of outsourcing software testing?

Outsourced testing models provide the benefits expected in any outsourced IT service model as well – such as access to a large pool of dedicated professionals with a wide range of skills, increased efficiencies due to the value creators & solution accelerators developed based on the teams' deep expertise & experience.

5. Which customers are most likely to outsource testing and why?

- Organizations experiencing increased quality issues owing to insufficient testing or ineffective processes and techniques
- Organizations witnessing rising costs related to quality. This may be due to too much effort spent on testing or may be due to business analysts and SMEs performing testing. Also, organizations that have built a huge mass of contractors over a period of time for testing may prefer to opt for outsourced testing as a measure to reduce cost.
- Organizations wanting to experiment with outsourcing. They typically tend to outsource testing rather than any other services because of the high cost-savings and relatively low risk involved.
- Customers with large integrated application environments requiring regular maintenance in the form of bug fixes and enhancements
- In the case of large applications, typically enterprise implementations, the support from vendors would cease after implementation and stabilization. In such scenarios, the client tends to look at testing service providers instead of building an internal QA team.
- Customers who do not have a dedicated QA organization where developers or business analysts perform testers' role. Outsourcing testing helps them reduce the money spent on developers or free up business analysts.
- Organizations planning a change in technology or a change in the business process. In-house expertise, especially subject matter experts, would be required for re-aligning with the new processes and systems. As a result, outsourcing everyday business frees up these critical resources.
- Organizations looking to optimize cost and improve coverage through automation. For them, building a test automation team internally may be a costlier option.

6. What are the range of savings that customers are realizing as they outsource portions of their internal QA requirements?

Testing application functionality requires good understanding of the business domain and hence organizations used business analysts or subject matter experts to validate the application's suitability to business. Scarce availability of skilled testing professionals and domain experts to support testing has been one of the greatest challenges faced by IT organizations. The emergence of software testing service providers helped business reap the following benefits:

- Cost optimization through efficient resourcing and utilization
- Capacity to address sudden spike and drop in demand, thereby eliminate overheads.
- Manage resource requirements based on need; especially while catering for the sudden demand rise & fall.
- Industry best practices and defined methodologies help to achieve higher quality.
- Testing tools expertise and infrastructure availability ensures reduced cost, improved quality and increased speed.
- Metrics-based reporting and management ensures transparency and helps to build trust.
- Continuous process improvements result in higher productivity and in increased cost savings.

Testing service providers invest significantly in various avenues of testing such as infrastructure, processes, people, tools, research, training and development etc. This helps their clients obtain higher value for the cost incurred, thereby maximizing the return on outsourcing.

Co-Shoring – All The Benefits Of Onshoring With The Competitive Costs Of Offshoring

by Osmar Higashi

Much has been said and practiced on outsourcing, with both results and market acceptance speaking for themselves. Limiting the discussion to the IT market, outsourcing is an alternative to otherwise hiring your own employees, aiming at advantages such as cost savings, ramping-up capacity for seasonal demands, and keeping energy focused in the organization core business. Subcontracting can also allow the entrance of specialized professionals ready to fulfill specific project needs, without dealing with issues such as the learning curve, head hunting, union relationship and many other concerns. Outsourcing is thus a reality.

Some known difficulties related to outsourcing are due to extra costs frequently not considered by the organizations. We can mention operational, cultural and linguistic differences, internal divergences, communication and knowledge transfer. Management issues must be considered as well, mainly if we keep our sight to geographically distributed teams. More project managers are needed due to a more complex process, even on just verifying the project activities' progress and results.

Different outsourcing solutions can be implemented, e.g., offshoring, nearshoring, onshoring, and co-shoring. Each one has its own advantages and issues that must to be evaluated according to specific customer needs. As a common feature we can point to the fact that the customer doesn't need to train/prepare employees or create a new specialty branch within its structure.

On the offshore modality, services are performed in foreign countries where it is possible to find a good balance between specialized workforce costs and enough maturity on the local market processes. Economical and political stability and linguistic skills are a must on this case.

Some countries are well established offshore sources, mainly India and China. Other countries are becoming more considered as serious contenders in this market, including other Asian countries, Latin American and Eastern Europe countries (source A.T. Kearney Global Services Location Index, 2007).

The great concern of the developing countries is to become good offshore options; but there are some constraints. It is not enough to be financially attractive, have skilled professionals and have high levels of internal experience. Great effort must be expended on the constant improvement required to stay competitive. As the competitors are making the same efforts, it is necessary to exceed the market's expectations constantly to stay on top.

We can figure out that this is developing to a stalemate due to the increasing wage costs for equivalent positions on the developing countries. As they are getting closer to those in the already developed countries, competitiveness tends to be considerably reduced and disadvantages start to arise.

Nearshoring is considered as an offshoring in the neighborhood, where cultural differences are minimal and time zone problems can be neglected. This results in the mitigation of some inherent offshore difficulties, including but not limited to service time and availability. Trip costs are considerably reduced, making management, adjustments and knowledge transfer much easier. This outsource model plays a very important role on foreign politics, creating new technology centers and bringing economic growth to the region.

As an important distinction, onshore work force is allocated within the customer country's borders. Obviously, the characteristic offshore issues relating to cultural and linguistic differences and lack of specialization or environment maturity do not apply. Some issues inherent to outsourcing like organizational culture differences and management problems can still be felt. Onshore can benefit from small-scale cities and countryside, where living is significantly cheaper and makes wages lower compared to other highly urbanized areas. Even though there are wage differences, these aren't very significant when compared to those that can be found abroad.

An onshoring stimulus arises from the global crisis effects, due to the need to stimulate some local economical growth at the developed countries. It is possible that internal policies help onshoring, encouraging the contracting of local specialized personnel.

Finally, the co-shoring model, an on- and offshoring hybrid, is intended to achieve the offshoring benefits, but keep service levels more close to onshoring, regarding communication, management and quality. Some corporations offer services merging a local branch (onsite or onshore) with remote structures (near- or offshore). The offshoring issues should be treated internally to the contracted organization, by their own distributed teams. Once the involved teams belong to the same organization, communication must be kept at the highest possible level, under the organization policies. Co-shoring costs can still be competitive because a very significant part of the service is performed by the offshore branch, bringing cost advantages when compared to those achieved with a purely onshore model.

Nowadays, we foresee the possibility of offering software testing services through Test Factories or independent testing centers which take advantage of the co-shoring trend. Organizations with internal software development, as their core business or not, must take into

3 day testing course by Alon Linetzki

May 04-06, 2009 in Brussels, Belgium

Limited places

Adding Business Value & Increasing ROI in Testing

Introduction
We are facing today an increase demand from test managers to improve productivity and product quality, to reduce costs of testing, to reduce resources, and to make-things-right-the-first-time.
This challenge is forcing test managers to improve in 3 major topics:

- Improve testing processes
- Manage testing project risks better
- Manage professional testers, and the supporting environment better by improving communication, diplomacy and negotiation skills

The course is enforcing those and aiming to educate test managers in how to be better in these areas of concern. The course includes practical exercises and simulations in the relevant topic areas.

Description
The learning objectives of Adding Business Value & Increasing ROI in Testing course focus on the below main areas of practical doing:

- To be able to know what test process improvement and TPI® model are
- Discuss a method for evaluating process maturity from different aspects
- Discuss the different dependencies between the testing processes
- Know how to quick-start TPI® effort in a testing organization
- Discuss which projects to pick for the TPI® pilot (with best chances to succeed)
- Understand the implications of managing risks in testing projects
- Understand the concepts of Risk Management
- Describe Risk Based Testing principals
- Understand what is the Risk language
- Define where RBT can assist during the testing life cycle
- To learn how to exercise an **excel tool** for risk strategy (planning phase)
- Discuss test execution strategy issues and RBT
- To learn how to exercise an **excel tool** for risk analysis (scheduling and execution phases)
- To learn about good communication concept and methods
- To know different paradigms of personalities, and reflect those on us
- Understand how to convince others without getting resistance
- Discuss how to present and pass a message to others
- Discuss how to give and obtain feedback, and get a positive impact

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

1450,- EUR
(plus VAT)

testing experience
Knowledge Transfer

Díaz Hilterscheid

account the well-known importance of independent test teams to achieve agreed software quality levels. Test factories don't mean the same as business process outsourcing (BPO) because even with testing activities being lead by an independent test center, the overall test are intrinsically linked to the software development. The specialized services offered are strongly related to the customer's activities. It is possible to deliver great advantages through the co-shore services offer.

With this contracting model, the local team can be formed by local and foreign professionals, providing a much-needed mix of profiles. This small and specialized team would be the interface with the customer, gathering the requirements, dealing with scope and scheduling, and presenting the test results. The local branch would deal with possible project changes and attend to customer satisfaction. This interface team, with strong communication and technical skills, would also be present at the offshored team interface, clarifying customer needs to the test team at home. The local and remote teams share the same corporate culture which helps minimize possible management and communication issues and enables them to be resolved internally and in a transparent way to the customer. With offshored services all customer relations are handled by the local branch, creating the illusion (sensation) of an entirely local service. Most of the cost would be generated by the offshored team, which composes the majority of the allocated resources.

Co-shoring means more than allocating specialized resources; it brings a service model which is internally structured and organized. Cultural and work issues are dealt by the supplier organization, and customizing is easier to perform, as well as tracking the project flow, gathering the expected results, and collecting lessons learned.

The organization willing to contract outsourced services must carefully analyze its reasons and benefits for choosing among the many described outsourcing options. By correctly balancing pros and cons, a significant part of the project success can be achieved. The co-shore model can address the market needs, offering a kind of transparent offshore and low-cost onshore.

Biography

Osmar Higashi is a senior IT consultant, with expertise in Software Testing and Quality Assurance. He holds a bachelor degree in Computer Science and MBA in Management.

In the last 15 years, Higashi is the executive director in charge of the strategies of RSI Informática (www.rsinet.com.br), a Brazilian leader in Software Testing and QA services. He had formed a team with many ISTQB Certified Testers and experienced consultants, to offer innovative and specialized services for the IT market.

He is the BSTQB (Brazilian Software Testing Qualifications Board) president, and continuously stimulates the software testing in Brazil and South America. He frequently participates in testing events and writes local papers. He also is member of ABRAMTI, a Brazilian non profit organization for IT improvement.

Quality, Excellence and Cost Effectiveness

– Outsourcing and Testing Centers in the "Silicon Wadi"

by Chen Bressler

It is common knowledge that these days many companies are looking for ways to save costs, and ROI figures are constantly being presented at board meetings. In this article I will try to show that a country which is leading in software development can also provide a cost-effective solution for outsourcing. In the following you will find a number of statistics gathered from a brochure published by The Israeli Export & International Cooperation Institute:

- **With more than 3,000 companies, Israel has the highest concentration of hi-tech companies in the world, second only to the Silicon Valley.**
- **After the Unites States and Canada, Israel has the largest number of NASDAQ listed companies.**
- **Israel has the highest ratio of university degrees to the population in the world.**
- **Israel leads the world in the number of scientists and technicians in the workforce - 145 per 10,000 (as opposed to 85 in the US, 70 in Japan and 60 in Germany).**

Israel is a major player in the world software market. Companies like Amdocs, Check Point, Comverse, Mercury and Nice, who are responsible for pioneering software solutions, have given Israel and its software market a good reputation, thanks to innovativeness, creativity and advanced technologies.

World software giants, such as IBM, HP, Microsoft, Oracle and Sun have long discovered the potential which lies in the Israeli market, and have established research and development, operation and production centers in Israel. The success of the local software and startup market, gave Israel the "Second Silicon Valley" / "Silicon Wadi" nicknames. All this in a country whose population is only 7.5 million!

The local software market symbolizes Israel's abilities and qualifications in the "knowledge-based" economy. A skilled workforce, combined with a competitive working environment and international quality standards have made Israel into a global leader in IT solutions and services.

Therefore, it is not surprising that the demand for quality developers and testers, who command the newest technologies and methodologies, is one which cannot be taken lightly in the competitive and export-based Israeli industry.

Today we can see a tendency of increase in the number of companies acquiring outsourcing services from Israel. Two trends in the market are the main reasons for this. The first reason is the effective costs of the Israeli work force, which is considerably lower than those in Europe and North America, while the costs in India and Eastern Europe are constantly rising. Additionally, the expected output from Israeli employees and the quality of challenges they can handle are very high.

According to a study published by the Gartner research group, Israel is among the 30 leading countries in outsourcing services. The study also relates to additional attributes - "Countries such as Ireland, Israel, Northern Ireland and South Africa fared well for language skills, because of the quality and quantity of English-language speakers."

For these reasons we can see a tendency where European and North American companies turn to acquiring software solutions and services from Israeli companies, specifically in testing and R&D projects which require skill, expertise and advanced and relevant knowledge.

An additional reason is that throughout the past years new models of outsourcing have developed in Israel in response to the need for the reduction of costs in the competitive market, and as an effective alternative for outsourcing in other countries. The models are based on the vast knowledge accumulated in the high-tech industry over the years, and on the large variety of population groups in Israel – the "melting pot" of cultures and populations. The combination of these elements is very significant in Israel, where one can find population groups living side by side, Jews with Arabs, immigrants from Russia, the US and Ethiopia with native Israelis, and secular with religious and ultra-orthodox.

The uniqueness of this model is that it enables a significant reduction in costs, while preserving quality and production. In most cases, the ideal solution is based on teams of experienced managers leading teams of rising talents drawn from this variety of population groups.

For example, in recent years a number of Israeli companies offer outsourcing solutions using people of the Haredi (or ultra-orthodox) community.

In Israel, the Haredim aged 20 and up make up about 9% of the Jewish population. The high birth rate in the Haredi community (7.7 births per mother), together with the lack of a general education, lead to a fairly high poverty rate. The employment level in this population group stands at a mere 33%. Lately, more and more projects, aimed at integrating some Haredim into the workforce, have been established. Many of the participants are happy for the opportunity given to them to "fit in" and utilize their potential.

Today in Israel, there are a number of projects which employ Haredi people in development and testing centers, while providing outsourcing services. In some cases, these workplaces employ only women. This may happen in centers located in cities which have a relatively high Haredi population. Most of these women that are employed are the sole breadwinners of their family, where their husbands dedicate their days to studying the Torah at a Yeshiva.

Other projects, such as the "ETGAR" (Challenge) program, working in coordination with the Israeli Ministry of Industry, Trade & Labor, turn to the Haredi male population. In this project, Yeshiva students who are appropriate

for the project in terms of learning skills, logic qualifications etc. are located and trained up for approximately one year of intensive studies of the most advanced technologies and methodologies to become software testers and developers.

The basis of the project is the desire to provide a solution with an optimized cost-effectiveness balance. In the STE – Software Testing Engineer – course, the students take the **ISTQB** exams and complete their studies with the CTFL certification.

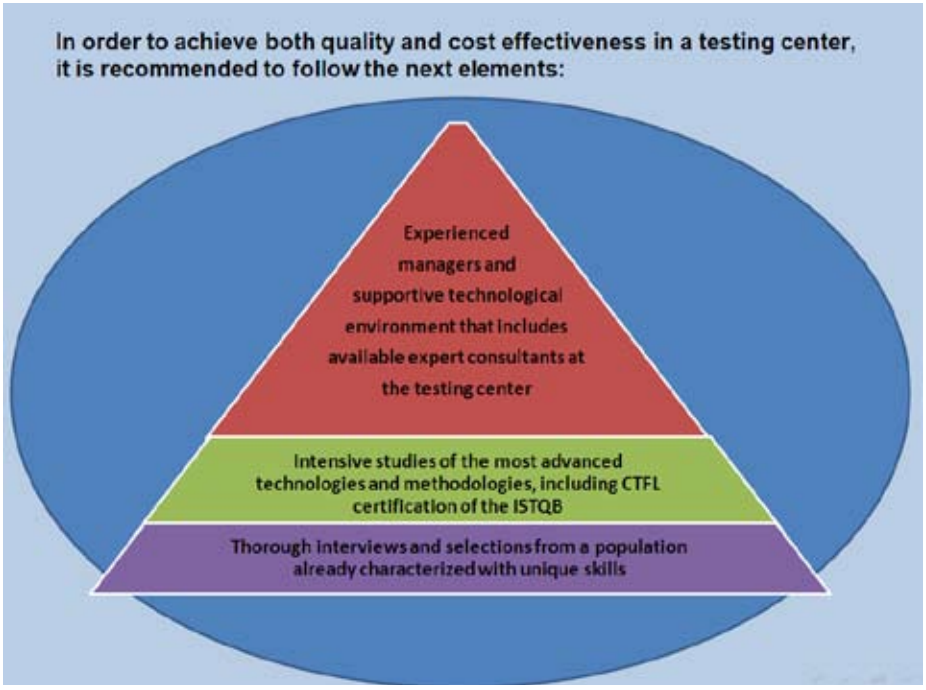
Following this, the course graduates work as software developers and testing engineers in designated development and testing centers, or they may join existing teams in high-tech companies working in an outsourcing context. Among the companies who already are successfully using these services, you will find **Microsoft**, **Qualisystems**, and **Comverse**. To quote a senior QA person: "We incorporated the graduates of these courses into the various divisions of our company, with a rate of success which is much higher than any of our expectations. It must be noted that it was easy to integrate these graduates immediately, and one could see they learned all the basics needed for joining the high-tech market, be it at the professional level or regarding the personal and interpersonal skills needed for workers in this demanding environment."

Besides the fact that only the elite are chosen for the projects, after going through thorough interviews and selections (less than 10% of candidates are accepted), the backgrounds of the students, namely Yeshiva education, contribute much to the development of their logical thinking. The reason for this is the style of Talmudic studies in Yeshivot, most of which teach the Talmud "in-depth". These studies require careful scrutiny of the text, paying attention to small details, and maintaining certain chains of thought.

A lecturer at a testing course gave me his impression of his students. "Throughout the course, I was very impressed by the remarkable abilities of the students. Years of Torah study have developed many of their important qualities, such as meticulousness, detailing, and searching for hidden meanings in every activity. These qualities are essential for a software tester, and these boys attain levels which I have rarely seen in all my years in the industry. Additionally, their high motivation and wish to join the industry is quite evident. They devote all their energy to these studies and the results are apparent."

During the course the students learn theoretical and practical material, which is focused on and relevant to the needs of the industry. As indicated by Neria Cohen, a student of the software developing course: "This course enables people like us, who have high learning abilities, to benefit from practical training and to learn much of the technology's best material in a relatively short time. If I may try to be objective – I believe the project fills a void in the Israeli society. This way, the industry benefits from talented people who can achieve new heights, and the project enables individuals, whom until now weren't able, to join the workforce and make use of their full potential." His colleague, Yossi Harel, concurs, "This project is a golden opportunity which enabled us to change our lifestyles and our way of thinking. We received a great gift here."

The utilization of the potential and abilities of the courses' graduates can be further illustrated by looking at Microsoft's blogs, where the elite list of bloggers was enhanced by the addition of Shlomo Goldberg (graduate of the software development course; blogger of the month – December 2008) and Baruch Frei (graduate of the software testing course). These examples demonstrate just how much learning abilities and motivation can be found in populations that previously had hardly been incorporated into our society's employment circle.

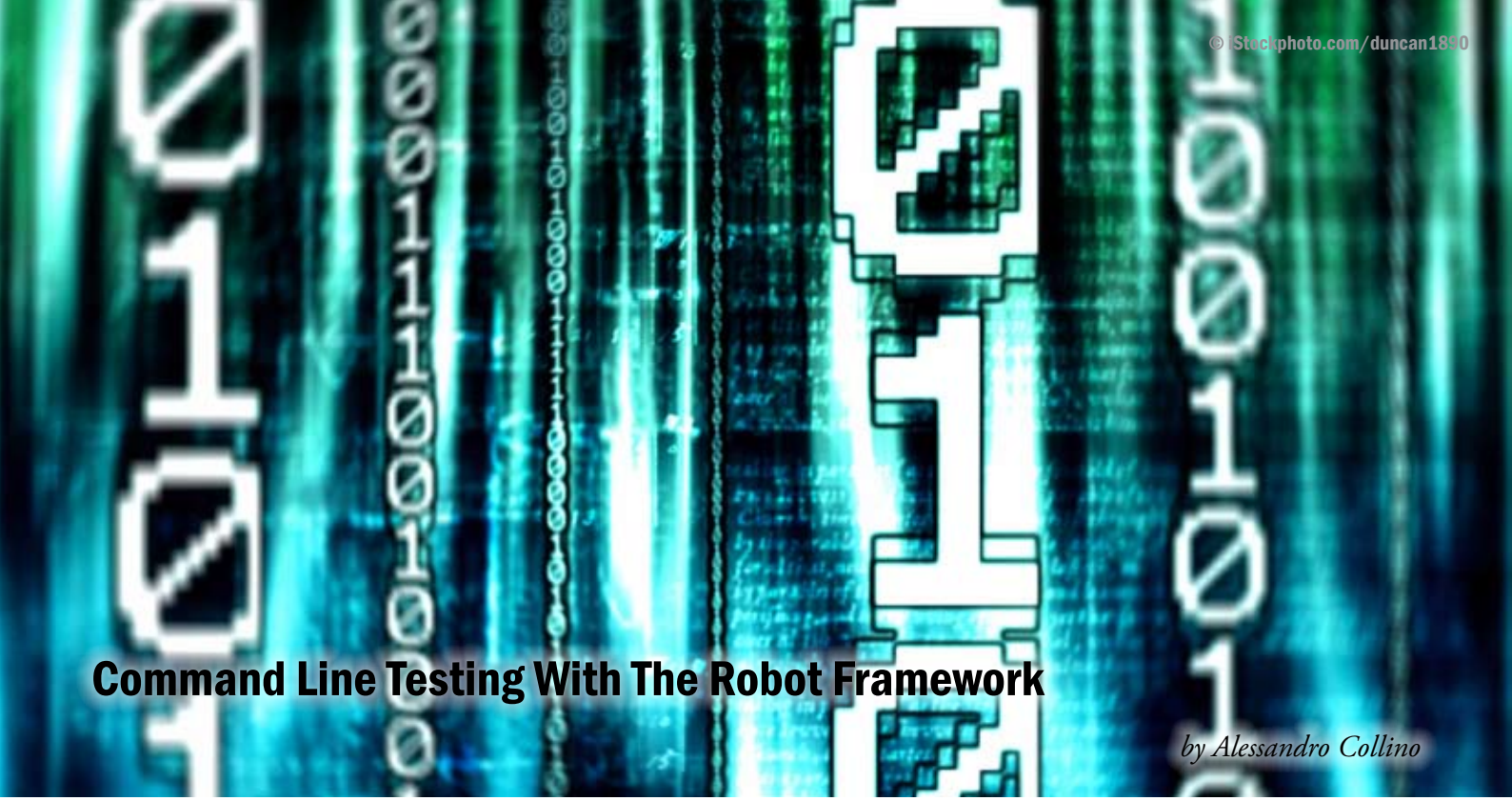


Summary

In an attempt to foresee the future, it is fair to assume that the use of outsourcing in the global high-tech market will continue to rise, even when we will see, with a sigh of relief, the end of the current financial crisis. Therefore, looking ahead to the next couple of years, we can expect the appearance of more new models of training and employing workers, turning to additional populations, and a flurry of changes in the countries who lead the table of outsourcing.

Biography

Chen is a Project Manager at SELA Group, the leading knowledge center for the High-Tech and IT industry in Israel. Among her responsibilities, Chen co-leads projects of outsourcing and testing centers and is responsible for international marketing of SELA's originally developed advanced courses and consulting services.
In her former position, Chen was a Project Manager at Sunrise Projects, a consulting firm in the field of R&D grants in Israel. There she specialized in raising millions of Shekels in grants for high-tech companies, and in business development in the EU Grants (FP7) department, which included initiating and creating strategic partnerships with high-tech companies and other groups in Europe.
In the six previous years, Chen established and managed the QA department of the Intellinx (formerly Sabratec) company, which manufactures a unique and innovative anti-fraud software product.
Chen holds a BA in computer science and mathematics, and a MBA (cum laude) with specialization in International Administration.



Command Line Testing With The Robot Framework

by Alessandro Collino

The Robot Framework [1] is a Python-based keyword-driven test automation framework for acceptance level testing and acceptance test-driven development (ATDD). It provides an easy-to-use syntax for creating test cases and is usually applied to three main domains: command-line, web, and GUI testing. For an effective introduction to the basics of Robot, the reader may refer to the article contained in a previous issue of Testing Experience [2].

The aim of this article is to discuss the practical use of Robot in one of the domains for which Robot offers powerful and flexible features: command-line testing.

The library interfaces for command line testing are essentially three:

- OperatingSystem Library
- Telnet Library
- SSH Library [3]

The article will focus on the SSH Library, that enables the execution of commands on a remote machine over an SSH connection and provides file transfer capabilities to and from a remote machine. The SSH Library works with both Python and Jython interpreters, but for our discussion we will only refer to the former.

Two practical aspects are going to be considered:

- How to use the SSH Library, through examples of increasing difficulty;
- How to use the Robot Framework in order to take advantage of the portability features that can be implemented through the use of the so-called "variable files".

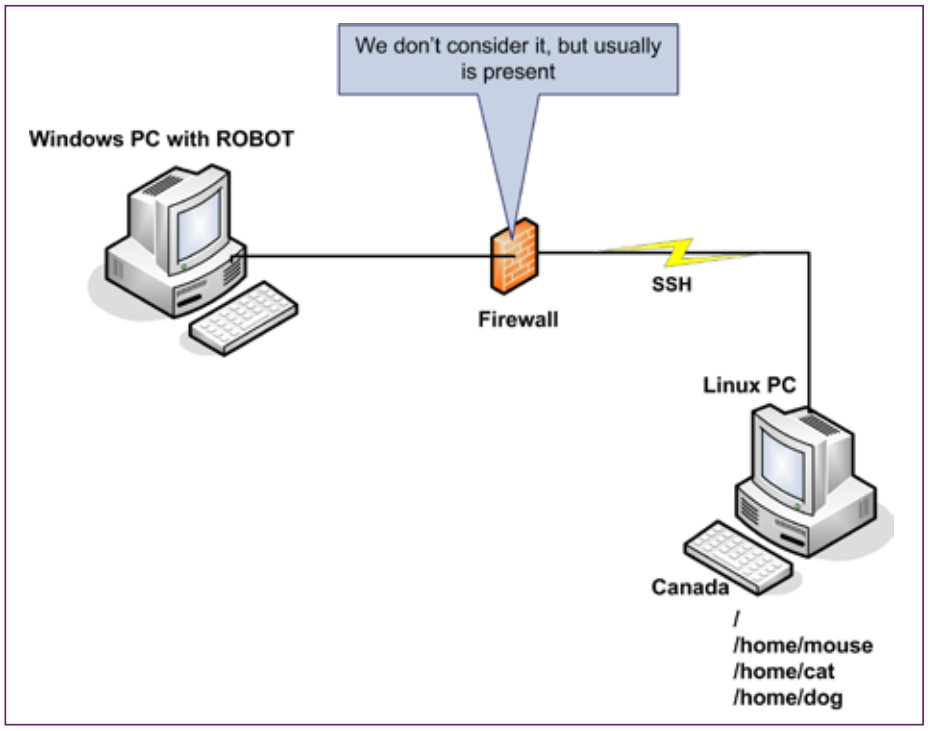


Figure 1

Getting Started

We assume that both Python and Robot are installed on a Windows system. Before using the SSH Library, PyCrypto [4] and Paramiko [5] shall be installed on the same system.

For Paramiko, installation instructions usually indicate that MinGW is needed. As a matter of fact MinGW is not needed for a successful Paramiko installation. PyCrypto requires a compatible C-compiler (such as MinGW, which comes with CygWin). Luckily you can just install the prebuilt version (no compilation needed).

Then get the Paramiko-zip, unzip the files and run 'setup.py build' first and then 'setup.py install'.

Once you are done with this, you will be able to use the SSHLibrary. Remember that these libraries should be installed in a place accessible by Robot (e.g. if Python25 has been installed in C:\Python25, then Robot must be able to access C:\Python25\Lib\ or C:\Python25\Lib\site-packages\).

First Example: Ssh Connection

To begin with our discussion, imagine we wish to run a test that connects to a remote machine, performs a simple "ls" command, and gets the

output on the standard output before closing all connections. Please refer to Figure 1.

We can implement this simple test case as follows:

Setting	Value	Value	Value	Value
Library	SSHLibrary			

Variable	Value	Value	Value	Value
${IP ADDRESS}	canada			
${USERNAME}	mouse			
${PASSWORD}	Hyu_123			
${COMMAND}	ls			

Test Case	Action	Arguments	Arguments	Arguments	Arguments
TC1	Open	${IP ADDRESS}			
	Login	${USERNAME}	${PASSWORD}		
	${out}	${err}=	Execute Command	${COMMAND}	both
	${out}=	Read Command Output			
	Close All Connections				

If we just want to see the results in the Robot log, then the execution of 'Execute Command'-keyword provides the information gathered from the 'ls'-command.

It is stored in the '${out}'-variable used in the above test case. In this way we should already be able to see in the Robot-log what is the value of ${out}, but we could also modify the test case so that the value is logged.

If we have to use 'Start Command', we will use 'Read Command Output' to read what has happened because 'Start Command' doesn't return anything.

The basic idea of 'Start Command' is that sometimes we need to run a command (for example to start a TTCN-tester simulating a network element) and immediately switch connection and run another command over another connection. 'Execute Command' would wait until the command is finished; in the TTCN-tester example, the TTCN-tester would not be finished until it gets some messages from the system under test, and those messages would not be coming unless triggered from the system under test through its GUI.

The modified test case looks like the following:

Setting	Value	Value	Value	Value
Library	SSHLibrary			

Variable	Value	Value	Value	Value
${IP ADDRESS}	canada			
${USERNAME}	mouse			
${PASSWORD}	Hyu_123			
${COMMAND}	ls			

Test Case	Action	Arguments	Arguments	Arguments	Arguments
TC1	Open	${IP ADDRESS}			
	Login	${USERNAME}	${PASSWORD}		
	${out}	${err}=	Execute Command	${COMMAND}	both
	Log	${out}			
	Close All Connections				

Second Example: Double Ssh Connection

The previous test case establishes a connection via SSH to a remote machine, performs a simple "ls" operation, and logs the result in ${out}.

In this second example we do the same as in the previous test case, but we also connect to another second machine, perform a simple "ls" operation, and log the result in ${out}. Please refer to Figure 2.

In order to be able to take another SSH connection from the first machine, in which we have logged in, some special arrangements are needed.

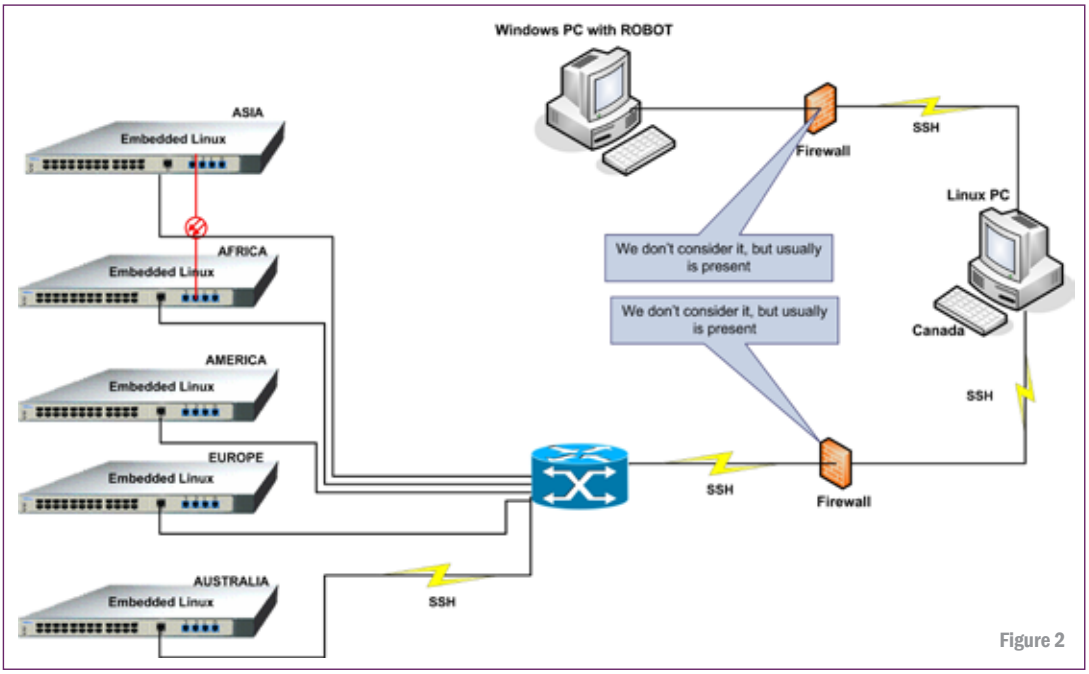


Figure 2

Setting	Value	Value	Value	Value
Library	SSHLibrary			

Variable	Value	Value	Value	Value
${IP ADDRESS1}	canada			
${USERNAME1}	root			
${PASSWORD1}	admin			
${COMMAND1}	ls			
${IP ADDRESS2}	australia			
${USERNAME2}	root			
${PASSWORD2}	admin			
${COMMAND2}	ls			

Test Case	Action	Arguments	Arguments	Arguments	Arguments
TC2	Open	${IP ADDRESS1}			
	Login	${USERNAME1}	${PASSWORD1}		
	${out}	${err}=	Execute Command	${COMMAND1}	both
	Log	${out}			
	${out}	${err}=	Execute Command	ssh ${USERNAME2}@${IP ADDRESS2} ${COMMAND2}	both
	Close All Connections				

In particular, we need to enable public-key authentication between those two machines so that we do not need to provide a password when executing a command. For info about the "public-key authentication", please refer to [6].

The test case can be implemented as follows:

```
ssh-keygen -q -f ~/.ssh/id_rsa -t rsa
```

Before running this test case, as described above, we need to enable public-key authentication. In our test case we try and log in as 'root', so the

```
[root@canada ~]# cd .ssh/
[root@canada .ssh]# ls -ltr
total 32
-rw-r--r-- 1 root root 8811 Dic 9 10:51 known_hosts
-rw-r--r-- 1 root root 241 Dic 12 11:25 id_rsa.pub
-rw------- 1 root root 951 Dic 12 11:25 id_rsa
```

following instructions are for a 'root'-user.

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

On the host (in our case: *canada*), run the following command (it requests the passphrase, but leave it empty):

This creates two new files under /root/.ssh-directory:

Copy the "id_rsa.pub"-file to the host we want to connect to (in our case: australia), for example in the root's home directory and run there the following command:

After that, we should be able to login and execute commands over SSH without passwords. In fact, we can observe that the variable ${PASSWORD2} is not used in the test case.

In our test cases we always need to use a syntax similar to that in the previous test case:
`"ssh user@somehost command".`

Third Example: Double Ssh Connection With Multiple Commands

In this example we execute a test case that extends the previous one, giving us the opportunity to execute on the target machine a list of useful commands.

This can be obtained in several ways. For the moment we do not consider the use of keywords, because it is more natural to use variables as commands. In this context we propose two solutions.

```
suite_variables.py
LIST__Commands=["ls", "cd opt/", "./cli -par1 120 -par2 125", "./cli Info -type PAR1"]
```

Setting	Value	Value	Value	Value
Library	SSHLibrary			
Variables	${CURDIR}/suite_variables.py			
Test Teardown	Close All Connections			

Test Case	Action	Arguments	Arguments	Arguments	Arguments
TC3	${conn1} =	Open and Login	@{Host1}		
	:FOR	${Command}	IN	@{Commands}	
		${out}	${err}=	Execute Command	ssh ${USERNAME2}@${IP ADDRESS2} ${Command}
	...	both			
		Log	${out}		

Keyword	Action	Argument	Argument	Argument	Argument
Open and Login	[Arguments]	${Host}	${User}	${PassWord}	${Alias}= ${User}
	${ConnId}	Open	${Host}	${Alias}	
	Login	${User}	${PassWord}		
	[Return]	${ConnId}			

1st solution : FOR-loop + "suite_variables.py" file

The test case proposed below uses the file "suite_variables.py".

The content of this file is the list of commands that we want to execute on the target machine. In our case, it looks like the following (where "cli"

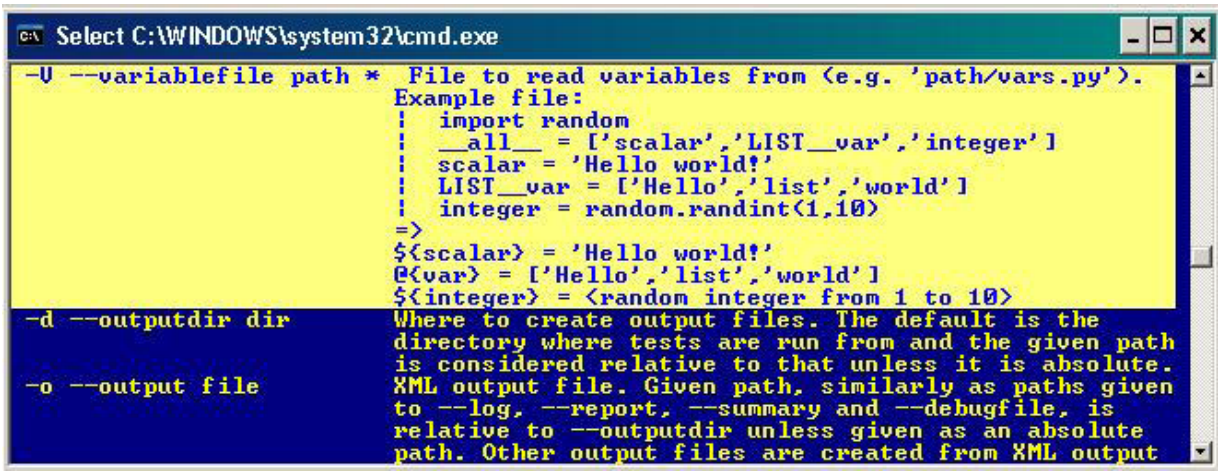


Figure 3

is simply an executable with two mandatory options):

The test case is the following:

```
LIST__Host1=["canada","root","admin"]
IP_ADDRESS2="australia"
USERNAME2="root"
```

The file "env_setup.py" contains information about the testing environment such as IP-addresses and user credentials. We shall run "env_setup.py" in the command line when we launch Robot (e.g.:"pybot -V env_setup.py testcasename.html").

Running on the host side the command "pybot –help", we obtain a very long list of options. In Figure 3 the output of the –V option is shown:

In our case, for example, the content of this file is the following:

env_setup.py

The idea behind this is that we can easily change our SUT (Software Under Test), if needed, by simply creating another "env_setup.py" that has the IP-addresses and other data pointing to some other SUT; we do not need to make a separate test case just for testing in a different environment. This increases *portability*, allowing also an easier *repeatibility*.

Another important feature of this test case is the definition of a keyword through the keyword table. The keyword is named "Open and Login" and is used as in column "Action" of the test case table exactly as the other keywords. This kind of keyword is named **USER KEYWORD** (briefly, a USER KEYWORD is used to combine a bunch of keywords, so that they can be used like a single keyword), but it does not have a correspondent keyword-handler function contained in a python library (in fact we do not import any Library into the Setting Table).

This type of keyword must not be confused with the **CUSTOM KEYWORD**, which, to be defined, requires instead a keyword handler in a python library that shall be imported into the test case through the Setting Table (typically written in Python language, even if Robot supports other languages encapsulated in a python library). Another observation, perhaps obvious at this point, is that we do not need any element in the keyword

```
./cli -par1 121 -par2 126
./cli -par1 122 -par2 127
./cli -par1 123 -par2 128
./cli -par1 124 -par2 129
./cli -par1 125 -par2 130
./cli -par1 126 -par2 131
./cli -par1 127 -par2 132
./cli -par1 128 -par2 133
./cli -par1 129 -par2 134
./cli -par1 130 -par2 135
./cli -par1 131 -par2 136
./cli -par1 132 -par2 137
```

table for CUSTOM KEYWORDS. The definition of CUSTOM KEYWORD is addressed in the next example.

Fourth Example: Double Ssh Connection With Custom Keyword

At this stage we should be able to connect via a double SSH connection from the host machine running Robot to the target, and perform single commands on it. In the previous examples the commands have been made explicit as variables (either in the variable table or by means of the "suite_variables.py").

```
import urllib;
import os;
import re;
import string;

class pars_gen_lib:

 def com3(self, i_Init, i_End):
 """ Space for comments
 """
 commands=[]
                a = int(i_Init)
                b = int(i_End)
 for i in range(a, b):
 j = i+120
 k =i+125
command = "./cli" + " -par1 %d" %(j) + " -par2 %d" %(k)
 commands.append(command)
 return commands
```

Now we want to execute on the target machine the following list of commands:

Obviously, it is not practical to invoke these commands on the target by using single variables. In this very simple situation we can observe that an invocation differs from the previous one only for having a VPI and a VCI incremented by one. This is a simple algorithm that can be implemented in python through a keyword-handler related to a CUSTOM KEYWORD that we can call "Com3".

Moreover, there could be opportunities to test similar configurations in the future, so the introduction of the keyword-handler could allow future re-use. We define the following python library called **pars_gen_Lib**:

This **pars_gen_Lib** python library should be in a file with the same name, with extension .py and stored either in the same directory as the test case, or in the Python envinronment Lib directory: in this second case the file **pars_gen_Lib.py** should be saved in C:\Python25\Lib\ ; at this point we can use the keywords Com3 and Com4 simply by importing the related library in the Setting Table.

We can now write the following test case:

Setting	Value	Value	Value	Value
Library	SSHLibrary			
Library	pars_gen_Lib			
Test Teardown	Close All Connections			

Variable	Value	Value	Value	Value
${IP1}	canada			
${USER1}	root			
${PWD1}	admin			
${IP2}	australia			
${USER2}	root			
${Com1}	ls			
${Com2}	cd opt			
${Com4}	./cli Info –type PAR1			

Test Case	Action	Arguments	Arguments	Arguments	Arguments
TC4	Open	${IP1}			
	Login	${USER1}	${PWD1}		
	${out}	${err}=	Execute Command	ssh ${USER2}@${IP2} {Com1}	
	Log	${out}			
	${out}	${err}=	Execute Command	ssh ${USER2}@${IP2} {Com2}	
	Log	${out}			
	@{Com3Cms}=	Com3	1	12	
	:FOR	${Command}	IN	@{Com3Cms}	
		${out}	${err}=	Execute Command	ssh ${USER2}@${IP2} ${Command}
	...	both			
		Log	${out}		
	${out}	${err}=	Execute Command	ssh ${USER2}@${IP2} {Com4}	
	Log	${out}			

The above test case implements the following actions on the target:

1. execution of the “ls” command in the target root directory
2. execution of the “cd opt” command in the target root directory
3. from the destination directory, the execution of the following commands:

```
./cli –par1 121 –par2 126
./cli –par1 122 –par2 127
./cli –par1 123 –par2 128
./cli –par1 124 –par2 129
./cli –par1 125 –par2 130
./cli –par1 126 –par2 131
./cli –par1 127 –par2 132
./cli –par1 128 –par2 133
./cli –par1 129 –par2 134
./cli –par1 130 –par2 135
./cli –par1 131 –par2 136
./cli –par1 132 –par2 137
```

4. from the destination directory, the execution of the following command:

```
./cli Info –type PAR1
```

Steps 1, 2 and 4 are single commands and can be implemented with the related variables in the Variable Table.

Step 3, on the contrary, calls the keyword-handler Com3 with parameters 1 and 12: this generates a FOR-loop of the 12 commands in the structure evidenced in grey . As we can see from the first evidenced row, the first operation is used to fill the list **@{Com3Cms}=** by calling our custom keyword **Com3** with parameters **1** and **12**. As a result the list is filled with all 12 commands, and a simple FOR-loop operation is performed.

We can underline two aspects in this test case:

1. In the Setting Table we have inserted a row to import our "pars_gen_Lib" python Library;
2. In Robot we cannot use keywords inside another keyword! For instance, if we have a custom keyword called "XYU" that returns a scalar, it is not possible to write the following:

```
${out} | ${err}= | Execute Command | ssh ${USER2}@${IP2}
XYU | both
```

Instead, we can store our keyword in a variable (both scalar and list) and use that as a parameter in the cell that already contains a keyword; the correct syntax for the example is therefore the following:

```
${cmd}= | XYU |
${out} | ${err}= | Execute Command | ssh
${USER2}@${IP2} ${cmd} | both
```

3. All parameters read from the Robot test case table are treated as a string.We can get Robot to treat your parameters as integers such as ${3}, but it is good practice to convert parameters to the form in which they are required inside our custom keyword, as we have done in the last example in our Com3 keyword, directly in Python.

Conclusion

After trying several Open Source Test Automation tools, for this article I chose the Robot Framework for three main reasons:

- its excellent architectural design;
- its pure keyword-driven technique, more flexible than the approach used by other renowned tools such as FIT/Fitnesse, where different tables need different fixtures;
- its simple approach to execute data-driven tests.

How the Robot Framework can be used to execute data-driven tests is described both in Quick Start Guide [7] and User Guide [8].

Summarizing, we have analyzed, step by step, how to improve simple test cases, emphasizing the variable file approach, the difference between custom and user keywords, and the use of the most important features offered by the SSH Library.

As a matter of fact, the SSH Library offers many other keywords (as presented in [3]), and it can be combined with the Operating System Library to easily implement powerful acceptance test cases, possibly structured in form of hierarchical test suites for the command-line testing of complex network architectures.

References

[1] http://robotframework.org : Robot Framework Main Page

[2] M. Michalak, P. Laukkanen. "Robot Framework for Test Automation". Testing Experience, December 2008.

[3] http://code.google.com/p/robotframework-sshlibrary/ : Robot SSH Library

[4] PyCrypto sources are available at: http://www.amk.ca/python/code/crypto.html Windows prebuilt PyCrypto: http://www.voidspace.org.uk/python/modules.shtml#pycrypto

[5] Paramiko is available at: http://www.lag.net/paramiko/

[6] http://sial.org/howto/openssh/publickey-auth/ : General info on public-key authentication

[7] http://robotframework.googlecode.com/svn/trunk/doc/quickstart/quickstart.html#data-driven-test-cases

[8] http://robotframework.googlecode.com/svn/trunk/doc/userguide/RobotFrameworkUserGuide.html#workflow-tests-vs-data-driven-tests

Biography

Born in 1975 and graduated in Computer Science and Information Engineering at the "Politecnico" of Milan, despite of the 7+ years of experience, Alessandro has matured and exploited know-how in conducting various medium-sized and large projects for several companies in various fields, in particular:

- Avionics (working on very complex equipments for the most important Helicopter Consortiums)
- Aerospace (working on a part of the ground segment for the ESA Mars Express project)
- Energy (working in a project conceived to provide new services in the market of energy distribution)
- Industrial (working on consumer electronic products)
- Telecommunications (where he is currently working for innovative platforms for UMTS)

Concerning the above activities, he has worked on all the phases of a complete classical product life cycle, ranging from specification to acceptance testing. In this respect, he has concentrated his efforts mainly on verification and validation activities (e.g. in avionics he has participated successfully in the certification of an embedded system for a real-time application, according to the DO-178B standard).
He works in Italy at ONION S.p.A. (www.onion.it) and collaborates closely with Dr. Gualtiero Bazzana (CEO of Onion S.p.A. and founder of ITA-SQTB) for important initiatives in software testing.
Alessandro is also active as speaker at software conferences, as program committee member of important conferences, as author for testing magazines, and is very much involved in the Agile community.

Agile Specification Quality Control:

Shifting emphasis from cleanup to sampling defects

by Tom Gilb

Traditional Inspection is often uneconomic and ties up valuable staff resources. Shifting the emphasis from cleanup (that is, from identifying defects and then removing them) to merely *sampling* the defect level of specifications, produces significant benefits. It enables the *quality level of specifications* to be determined more rapidly. Consequently, the QC can be carried out more frequently. Systems and software engineers rapidly learn, through SQC feedback, to take standards seriously, which in turn reduces defect injection. Further, by analyzing where/how the defects occur, continuous process improvement can be supported.

Introduction

If we carry out inspection of specifications properly (Gilb and Graham 1993), the cost is barely tolerable for some: about one hour of effort, per page[1] checked, per systems engineer or software engineer. The harvest, even if we are skilled, is only to identify between 40-80% of the major defects. That leaves many remaining major defects undetected, and many of these *will* be found, at considerable cost, during testing or in the final released product.

Of course, finding defects using traditional inspection (and fixing them) earlier than the test stage is beneficial, and may even pay off. However, there is a better way: Agile Specification Quality Control (Agile SQC). It ought to appeal to all Spec QC purposes, and especially to the many organizations that have not been able to stomach the high costs, and low effectiveness, of traditional inspection.

The main concept of Agile SQC is to shift emphasis *from* 'finding and fixing defects', to 'estimating the specification defect density', and using this information to motivate systems and software engineers to learn to avoid defect injection in the first place. Such a shift permits a dramatic cost saving. When our QC purpose is *measurement*, rather than 'cleanup', we can *sample*, rather than have to check 100% of the specifications. This is the major opportunity that Agile SQC provides. The main *purpose* of Agile SQC is to *motivate* individuals to learn to reduce major defect insertion. Secondary purposes include:

- To prevent uneconomic major-defect density specifications from escaping downstream – and thus to avoid the consequent delays and quality problems. The major tactic to achieve this is to impose a numeric exit-barrier for the specification process, such as 'only a maximum of 1.0 remaining majors per page';
- To teach and reinforce current specification standards.

Process Details

Traditional Inspection Method: The old inspection method (widely practiced in CMM Level 3 as peer reviews) was based on the idea of inspecting 100% of all pages, at optimum rate checking (one page per hour), using a review team of between 2 and 5 software and systems engineers. The maximum yield of major defects from such an inspection process is in the range of 40%-80% depending on specification type (For example, a maximum of 60% for software source code specifications, and a maximum of 80% for requirements specifications – in practice, however, it is actually more likely that only 30% is achieved since malpractice is common). The reported ability to actually *correctly* correct major defects, once found, is only 5 out of 6 fixes attempted (Fagan 1986 reported in Gilb and Graham 1993).

All this amounts to the following:

- The same order-of-magnitude defects *remaining*, as before the quality control process was applied;
- Little or no change in the defect *insertion* density. In requirements specifications, this *regularly* exceeds 100 major defects per 300 lines of specification (personal experience by field measurement over many years).

New Agile SQC Method: The new 'Agile SQC method' is based on the following:

- *Sampling* of a specification;
- A few (1 to 3) pages at a time;
- Starting early (perhaps once the first 5% of a large specification is written);
- Frequently (every week or so) until the work is completed.

For each individual systems or software engineer (each one must be motivated and trained personally), their sampled specification pages will be checked against a set of a few simple rules – usually about 3 to 7 rules are applied (For example, for initial checks, these could be: *Clear enough to test, Unambiguous to intended readers, and No design options in the requirements)*. The reviewers/checkers are asked to identify all deviations from these rules. Any deviation is termed a 'specification defect'. The reviewers/checkers are then asked to classify any specification defect that can potentially lead to loss of time, or significant reduction in product quality, as 'major'. The entire checking session might use only 2 engineers for 30 to 60 minutes. This might seem quite a high checking rate, but remember that only a few rules are being used and no other documents are being consulted to check

1 A page is defined as 300 words of non-commentary text. Non-commentary text is core specification or background text; it is not notes or other commentary text.

out the original source of material, so we can go faster. In any event, as long as we turn up more defects than the threshold exit level for defects, then exactly how effective we are in detecting defects is a secondary issue.

The major defect findings are reported to a review leader, who calculates the estimated number of defects actually present, based on the total found by the team. An inexperienced team is usually about one third effective, so the estimated total number of majors per page is about three times the total of *unique* majors found by the team. This is a very rough calculation, but it seems to work well in practice.

A pre-arranged standard for exit control (the *fail to exit* level) is set for unacceptable specification major-defect density. Initially, it can be set at 'anything more than 10.0 majors per page'. In the longer term (beyond 6 months of culture change), the aim should be to set the limit at 'anything more than 1.0 majors per page'. To give some examples, IBM reported using a maximum of 0.25 major defects per page (Humphreys 1989). NASA reported a standard of using 0.1 major defects per page (Bhandari et al. 1994). The initial limit set is a matter of trying to get better as fast as humanly possible. Ultimately, it should become a matter of finding the level that pays off, for the class of work you are doing.

Note: There are several limitations to this simplified Agile SQC process:

- It is only a small sample, so the accuracy is not as good as for a 100%, or than for a sample which is larger than a few pages;
- The team will not have time or experience to get up to speed on the rules and the concept of major defects;
- A small team of two people does not have the probable greater effectiveness of 3 or 4 people;
- The entire specification will not have been checked, so there will not be the basis for making corrections to the *entire* specification;
- The checking will not have been carried out against all possible source documents (Usually in the Agile SQC process, no source documents are used, and memory is relied on. While this means that the checking is not nearly as accurate, it does considerably speed up the process).

However, if the sample turns up a defect-density estimation of 50 to 150 major defects per page (which is quite normal), that is more than sufficient to convince the people participating, and their managers, that they have a serious problem.

As discussed earlier, the immediate solution to the problem of high defect density is *not* to set about removing the defects from the document, because the same order of magnitude level of defects would still remain. The best solution for a document with a high defect density is to rewrite it *entirely*, using an individual who does not insert too many defects. Long term, the most effective practical solution is to adopt Agile SQC as part of the corporate process, and most importantly, make sure that each individual specification writer takes the defect density criteria (and its 'no exit' consequence) seriously. They will then learn to follow the rules; and as a result will reduce their personal defect injection rate. On average, a personal defect injection rate should fall by about 50% after each experience of using the SQC process. Widespread use of Agile SQC will result in large numbers of systems and software en-

Agile SQC Process

Entry Conditions

- A group of two, or more, suitable people* to carry out Agile SQC is assembled in a meeting.
- The people have sufficient time to complete an Agile SQC. Total Elapsed Time: 30 to 60 minutes.
- There is a trained SQC team leader at the meeting to manage the process.

Procedure

P1: Identify Checkers: Two people, maybe more, should be identified to carry out the checking.
P2: Select Rules: The group identifies about three rules to use for checking the specification. (My favorites are clarity ('clear enough to test'), unambiguous ('to the intended readership') and completeness ('compared to sources'). For requirements, I also use 'no optional design'.)
P3: Choose Sample(s): The group then selects sample(s) of about one 'logical' page in length (300 non-commentary words). Choosing such a page at random can add *credibility* – so long as it is *representative* of the content that is subject to quality control. The group should decide whether all the checkers should use the same sample, or whether *different* samples are more appropriate.
P4: Instruct Checkers: The SQC team leader briefly instructs the checkers about the rules, the checking time, and how to document any defects, and then determine if they are *major* defects (majors).
P5: Check Sample: The checkers use between 10 and 30 minutes to check their sample against the selected rules. Each checker should 'mark up' their copy of the document as they check (underlining issues, and classifying them as 'major' or not). At the end of checking, each checker should count the number of 'possible majors' (spec defects, rule violations) they have found in their page.
P6: Report Results: The checkers each report to the group their number of 'possible majors.' Each checker determines their number of majors, and reports it.
P7: Analyze Results: The SQC team leader extrapolates from the findings the number of majors in a single page (about 6 times** the most majors found by a single person, or alternatively 3 times the unique majors found by a 2 to 4 person team). This gives the major-defect density estimate. If using more than one sample, you should average the densities found by the group in different pages. The SQC team leader then multiplies the 'average major defects per page density' by the 'total number of pages' to get the 'total number of major defects in the specification' (for dramatic effect!).
P8: Decide Action: If the number of majors per page found is a large one (ten majors or more), then there is little point in the group doing anything, except determining how they are going to get someone to write the specification 'properly', meaning to an acceptable exit level. There is no economic point in looking at the other pages to find 'all the defects', or correcting the majors already found. There are simply too many majors not found.
P9: Suggest Cause: The team then chooses any major defect and thinks for a minute why it happened. Then the team agrees a short sentence, or better still a few words, to capture their verdict.

Exit Conditions

Exit if less than 5 majors per page extrapolated total density, or if an action plan to 'rewrite' the specification has been agreed.

Figure 1: Specification of the Agile SQC Process

Notes:
* A suitable person is anyone, who can correctly interpret the rules and the concept of 'major'.
** Concerning the factor of multiplying by '6 ': We have found by experience (Gilb and Graham 1993: reported by Bernard) that the total unique defects found by a team is approximately twice that of the number found by the person who finds the most defects in the team. We also find that inexperienced teams using Agile SQC seem to have about one third effectiveness in identifying the major defects that are actually there. So 2 x 3 = 6 is the factor we use (Or 3 x the number of unique majors found by the entire team).

gineers learning to follow the rules. To get to the next level of quality improvement, the next step is to improve the rules themselves.

Case Study 1: A Financial Organization

In 2003, a large multinational financial group was a pilot user of this Agile SQC process. It also had combined this with adopting a specification and planning language, Planguage (Gilb 2005). After six months, the organization reported the following for requirements and design specifications:

- Across 18 development projects using the new requirements method, the average major defect rate (per page) on first inspection is 11.2;
- 14 of the 18 development projects exited successfully on first pass Agile SQC. The other 4 development projects failed to meet the exit criteria of 10 major defects per page, the projects' specifications had to be improved, and were then re-inspected;
- A sample of 6 development projects with requirements in the 'old' specification format were tested against the following set of rules:
  - The requirement is uniquely identifiable;
  - All stakeholders are identified;
  - The content of the requirement is 'clear and unambiguous';
  - A practical test can be applied to validate delivery of the requirement.

The average major defect rate (per page) in this sample was 80.4.

A few months later, as a result of the continuing overall success of the pilot testing, the client decided to spread Agile SQC widely to all types of technical specification.

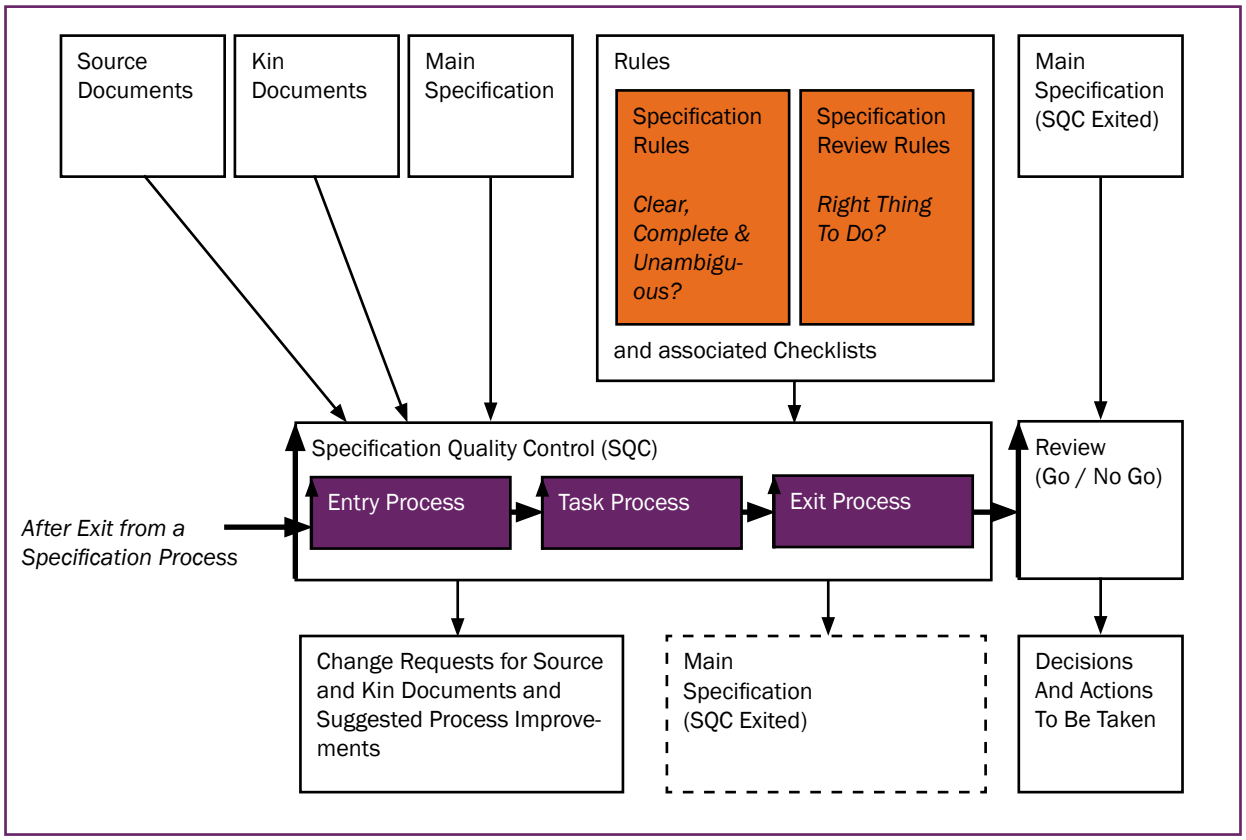


Figure 2. Overview of the SQC Process (Gilb 2005)

Case Study 2: A Jet Engine Manufacturer

At one of my clients, we sampled 2 pages of an 82-page requirements document: four managers checked page 81, and four other managers, who were directly involved with the requirement specifications projects, checked page 82. These pages were all 'non-functional' requirements (such as, security). We agreed to check against the following simple set of requirement specification rules:

1. **Unambiguous** to intended readership
2. **Clear** enough to test.
3. No **Design** specifications (= 'how to- be good') mixed in

Violation of any one of these rules constituted a specification 'defect' and was classified either as 'major' (likely to result in potential damage to effort or quality) or 'minor' (no way they can harm us, even though they are defects).

We also agreed a specification exit level of 'No more than one *remaining* major defect per page'. They 'agreed' (for demo purposes!) that any manager who signed off (approved) a requirements specification with more than 100 remaining major defects per page should be fired for incompetence. Later that day they themselves were, as we shall see, to provide clear numeric evidence that – they themselves should be 'fired'!

The 8 managers were given 30 minutes to check their page. At the end they reported the following major defects found by themselves individually:

Page 81 (three quarters of a page): 15, 15, 20, and 4 majors.

Page 82 (a full page): 24, 15, 30 and 3 majors.

Estimating the number of major defects found by the team

From the results of this input, we could estimate the number of unique major defects found by the **team**. First we had a hypothetical choice of either logging all the unique major defects (Using non-Agile methods, logging would take 3 minutes for each defect resulting in a 3 hour job), or estimating the result approximately. Not surprisingly, the managers chose the quick estimation. To estimate the number of unique majors (that is, the number of majors that are not duplicated - so if the same defect is found by more than one checker it only counts as one defect); we can double the count of the largest number of majors found by one individual in a small (2-4 people) group. This is based on observations done at Cray Research (Gilb and Graham 1993 pp. 299-301). From personal experience, it works well. In this case, this means that the group working on page 82 had about (2 x 30) 60 majors per page found (±15 majors of course). The group working on page 81 had about 40 totally unique major defects they could log if they so chose to log them in detail.

Estimating the total number of defects per page – including those NOT found by the team

Of course, inexperienced checkers do not find 100% of the major defects present - they find only about a third. Remember even experienced checkers carrying out *source code* inspections peak at a source code bug-finding effectiveness of 60% (Gilb and Graham 1993 reported by IBM MN), and most groups are not that good. Requirements and design checking tend to have an effectiveness ranging between 30% and 80% or more, depending on a wide range of process factors. These effectiveness factors include speed of checking, available related project data, use of standards and checklists, and intelligibility of the specification being checked.

Can we verify the level of checking effectiveness in practice?

If you want to prove these estimates, the proof is simple: carry out an inspection, and then remove the major defects you have identified. That should leave twice that number estimated remaining – the two thirds NOT found by checkers (In this example, 80 major defects for page 81, and 120 for page 82). This sounds incredible. How could people miss so many on a single page? The proof comes when you repeat the checking process, and predictably find one third of the remainder (one third of 80); and can prove they *were there* on the first checking pass. Skeptics turn into believers at this point. We have carried out this test on our courses for years, and it always proves the case.

So, how many major defects are there in total on the page?

In this case, the managers accepted my assertion – that the 60 majors on page 82 were an indication of about 180 majors in the page (and 150 majors on page 81, indicative of the same density as page 82). Now this indicates an average of (120 + 180)/2 = 150 majors per page. I asked the managers if they felt this was probably typical for the other ('functional') pages. They said they had no doubts that it was. If managers are skeptical, the solution is simple, take another sample at random. I can assure you that the result found for defect density will be essentially the same order of magnitude.

Then, how can we estimate the total number of major defects in the specification?

Now this leads us to an estimation that we have about 150 (average per physical page) x 82 (total pages) = 12,300 major defects in total. I was initially quite shocked when calculating this number. But the managers were for some strange reason not as skeptical as I was. I did not know anything about the project beyond that the requirements had just been handed to me 45 minutes earlier, and that the managers were somehow responsible.

How many bugs will be generated as a result of these specification major defects?

Let's carry on with the calculations! Now another factor that has to be taken into consideration is that not *all* major defects in specifications will directly lead to bugs. The problem being that we don't know exactly *which* of the major specification defects will *actually* cause bugs to be inserted - that depends on the 'sleepiness of the programmers on the day'! Two pieces of research I recall showed that 25% to 35% of the majors actually turn into bugs. For example, to make this plausible, a random guess as to the correct interpretation of an ambiguity with 2 options would give a 50% chance of a bug and 50% not. I have found that a good rule of thumb, that correlates well with observed reality, is that one third of the major defects will cause bugs in the system. So, for this example, that implies that about 4,100 (= 12,300/3) bugs will occur.

What do these major defects cost in project terms? How do they delay the project?

One of my clients (Philips Defence, UK, see case study in (Gilb and Graham 1993, page 315)) studied about 1,000 major defects found in specification inspection of a wide variety of systems engineering specifications. They determined that the *median* downstream cost of *not* finding the majors would have been 9.3 hours (range up to 80 hours). So I use 10 hours as a rough rounded approximation of the cost of a major if it occurs downstream (at test and field stages).

Well, in this case study, that implies 41,000 hours (10 x 4,100 defects that hit us) effort lost in the project through faulty requirements. I was quite shocked at the implication of this quick estimate based on a small sample. But the managers were quite at home with it. They responded, "Don't worry, Tom, we believe you!"

"Why?" I asked. So they explained, "Because (and we know you did not have any inkling of this) we have 10 people on the project, each using about 2,000 hours per year, and the project is already 1 year late (a total of 20,000 hours), and we have at least one more year of correcting the problems before we can finish."

Case Study 3: An Air Traffic Control Project (In Sweden & Germany)

Another client had a seriously delayed software component for an air traffic control simulator. The contract dictated about 80,000 pages of logic specifications. The supplier had written and approved about 40,000 pages of these. The next stage for the logic specifications was writing the software.

The divisional director, Ingvar, gave me the technical managers for a day to try to sort out the problem. These managers had each personally signed off the 40,000 pages. We pulled 3 random pages from the 40,000 and I asked the managers to find logic errors in the specifications – errors in the sense that, if coded, the ATC system would be wrong. Within an hour of checking, they found 19 major defects in the 3 sample pages. They agreed these pages were representative of the others.

That evening, Ingvar took 30 minutes to check the 19 defects personally, while his managers and I waited in his office. He finally said, "If I let one of these defects get out to our customer, the CEO would fire me!"

Now the 19 defects found in the 3 pages represent an actual defect density of approximately three times that (that is, they probably did not find two thirds of the existing defects). So the managers had signed off about (20 x 40,000) 0.8 million bugs. And they had only done half the contracted logic specification. Well, the sample told us a great deal.

We started thinking that afternoon about what could have been done better. The conclusion was that we had a 'factory' of analysts producing about 20 major defects per page of ATC logic specification. We also concluded that if we had taken such a sample earlier, say after the first dozens of pages written, we might have discovered the systemic defect-density pollution-rate earlier, and could have hopefully done something about it.

Too bad that they did not have Agile SQC! The project got completed; but only after being sold off to another corporation. The director lost his job, and it was not just for a single defect.

The irony was that when I first met the director, he told me he had read a book of mine. Too bad he did not practice what he read. His corporation, I later realized, had a bad ingrained habit. They did not review specifications until all pages were completed.

I asked the manager responsible for the third signature on the specification approval, why he signed off on what we all acknowledged was a tragedy. He told me it was because 'the other managers signed it ahead of him'. I guess that is when I lost faith in management approvals.

Agile Sqc Estimations And Calculations

At this point, it is worthwhile summarizing the overall Agile SQC process of estimating and calculating. See Figures 3 and 4, which show how to arrive at the defect level for a specification and how to calculate the number of remaining defects in a specification respectively. The calculations shown are for yet another case study.

only 10 places

Exploratory Testing Tutorial

by James Lyndsay

June 4-5, 2009 in Berlin

Getting a Grip on Exploratory Testing
Exploratory Testing is a discipline. Focussed on the just-delivered system, it exposes real-world problems more rapidly and more reliably than approaches which concentrate on prior expectations. It forms a powerful complement to scripted testing, and is especially powerful when matched with the large-scale confirmatory testing found on agile projects. Good exploratory testers find better bugs, and give fast, clear feedback to their teams.
Many testers are able to work without scripts and explore a system. However, most take a single exploratory approach, and so become less effective when they have exhausted that approach. Many teams use exploratory testing, but do not know how to manage it effectively.
This course allows participants to experience a range of exploratory techniques in a disciplined framework that will allow exploratory testing to be managed and integrated with existing test activities. The course is built around hands-on, brain-engaged exercises designed to enhance and reinforce the learning experience.

Participants will discover:
- The test design skills to probe a system and trigger a bug
- The analysis skills to model the system and understand a bug
- The discipline to manage their exploration and sustain their bug rate

Course Description
This two-day course is a comprehensive guide to exploratory testing, with exercises deigned to engage testers of all abilities, and structured workshops to help participants use and share the lessons learned. The course will introduce participants to a wide variety of approaches to exploratory testing – parsing the system, systematic exploration, modelling for success and failure, questioning, attacks and exploitations. Starting with three basic exploratory frameworks, we will construct models of the systems under test, and use those models to enhance our testing and to judge problems found. We will develop attacks and exploitations to reveal deep risks, and look at the potential risks in target technologies.
Throughout the course, we will use session-based testing to support the personal and team discipline necessary to support effective exploration, and will examine ways that ET can be managed within existing processes.
This tutorial will be of greatest use to test analysts, senior testers and test managers, but will also be immediately relevant to designers and coders. Direct experience in exploratory techniques is not necessary, but delegates with one or more year's experience of hands-on testing will get most from this course.

"I attended James Lyndsay's 2-day exploratory testing tutorial and it was very useful and fun.
If you expect loads of theory to be thrown at you - don't bother. But if you want lots of hands-on exercises, eye-openers and discussions, combined with a refreshing view on testing, this is the course for you. On top of that, James is a very engaging 'teacher', and an allround nice guy :-)"
Zeger Van Hese, CTG Belgium N.V./S.A.

te testing experience
Knowledge Transfer

1250,- EUR
(plus VAT)

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

Agile Specification Quality Control (SQC) Form - An Example Filled Out

SQC Date: May 29, 200X. SQC Start Time: _____
SQC Leader: Tom.
Author: Tino. Other Checkers: Artur.

Specification Reference: Test Plan. Specification Date and/or Version: V 2.
Total Physical Pages: 10.
Sample Reference within Specification: Page 3.
Sample Size (Non commentary words): approx. 300.

Rules used for Checking: Generic Rules, Test Plan Rules.
Planned Exit Level (Majors per page): _____ or less.
Checking Time Planned (Minutes): 30. Actual: 25.
Checking Rate Planned (Non commentary pages per hour): 2.
(Note this rate should be less than 2 pages per hour)

Actual Checking Rate (Non commentary words per minute): _____
Number of Defects Identified by each Checker:
Majors: 6, 8, 3. Total Majors Identified in Sample: 17.
Minors: 10, 15, 30.
Estimated Unique Majors Found by Team: 16 ± 5.
(Note 2 x highest number of Majors found by an individual checker)
Estimated Average Majors per Page: ~16 x 3 = 48.
(A Page = 300 Non commentary words)
Majors in Relation to Exit Level: 48/1 (47 too many).
Estimated Total Majors in entire Specification: 48 x 10 = 480.

Recommendation for Specification (Exit/Rework/Rewrite): No exit, redo and resubmit.
Suggested Causes (of defect level): Author not familiar with rules.
Actions suggested to mitigate Causes: Author studies rules. All authors given training in rules.
Person responsible for Action: Project Manager.
SQC End Time: 18:08. Total Time taken for SQC: _____

Figure 3. A example of an SQC form filled out

Estimating Remaining Major Defect Density

Assumptions:
A logical page (page) is 300 non-commentary words.
Your SQC effectiveness is 33.3% and your SQC is a statistically stable process.
One sixth of your attempts to fix defects fail (One sixth is average failure to fix).
New defects are injected during your attempts to fix defects at 5%.
The uncertainty factor in the estimation of remaining defects is ± 30%.
Probable remaining major defects per page =
'Probable unidentified majors' + 'Bad fix majors' + 'Majors injected'
Let E = Effectiveness expressed as a percentage (%) = 33.3%

If 33 major defects per page have been found during SQC.
Probable unidentified majors =
Major defects total estimated 3 x Found Majors (33) = about 100 ±30

Bad fix majors = One sixth of fixed majors =
Of 30 attempted fixes, 5 major defects in each page are not fixed.
This is useful to recognize.
Even if you found all defects, 1/6 would remain after all were fixed.

Majors injected = 5% of majors attempted to be fixed = 1.5 major defects per page.
(this is not always calculated, since it is small, compared to the error margin)

Probable remaining major defects/page, *after fixing what we found in a sample* =
66 (not found) + 5 (not fixed) = roughly 71 remaining major defects per page.

Taking into account the uncertainty factor of ± 30% and rounding down to the nearest whole number gives 50 Remaining Major Defects per Page
(Minimum = 50, Maximum = 92 remaining major defects per page).

Figure 4. An example of calculating the remaining major defects per page

Continuous Process Improvement

Notice how towards the end of Figure 3 there are two questions concerned with analyzing the origin of the defects (that is, 'Suggested Causes' and 'Actions suggested to mitigate Causes'). The aim of these questions is to identify problems in the work practices that need correction. This approach is identical to Capability Maturity Model Level 5, and to the Defect Prevention Process (see discussion of Mays in (Gilb and Graham 1993)).

In the Raytheon Study (Haley et al. 1995, Dion 1993), this process improvement effort reduced rework costs, within about 7 years, from about 27% of all development costs, down to about 4%. Before that happened though, the individual discipline of software engineers actually following their existing (bad) processes, led to a reduction, in a year, from 43% rework costs to the 27% cited above. So there is lots of short-term improvement available by getting people to follow even simple standards.

Personal experience with SQC is that by merely motivating people to follow the simple rules of 'clear/unambiguous/no design' in requirements, we can reduce the number of major defects inserted into requirements by, in one case an average of 80 majors/page to about an average of 11 majors/page within 6 months. Corporate engineering measurements (Douglas Aircraft 1988) and other examples indicate that the individual rate of reduction of defect insertion is about 50% per learning cycle. So, in about 7 cycles of writing specifications and measuring defects, an individual gets to the exit level of less than one major per page.

Summary

Agile SQC costs very little, but its effect on early control over injected defects is significant. It can drive defect injection down by one and then, with time, two orders of magnitude.

The key Agile SQC concept compared to traditional Spec Inspection methods is to measure by *sampling*, and use the information to *motivate* people to 'learn the rules' (that is, the standards and/or best practices), and *reduce their defect injection*.

Traditional Spec Inspection techniques are doomed to high costs and low effect because:

- they can only hope to find about half the problems (Given 40-80% is the very best in practice);
- they spend approximately 3-4 hours engineering effort *per page* of specification (for full effectiveness).

References

Bhandari, I., M.J. Halliday, J. Chaar, R. Chillarege, K. Jones, J.S. Atkinson, C. Lepori-Costello, P.Y. Jasper, E.D. Tarver, C.C. Lewis and M.Yonezawa, In-process improvement through defect data interpretation, *IBM Systems Journal*, Issue 1, Volume 33, page 182, 1994.

Dion, Raymond. July 1993. Process Improvement and the Corporate Balance Sheet. *IEEE Software*. Pages 28-35.

Fagan. M. E, 'Advances in Software Inspections', *IEEE Transactions on Software Engineering*. Vol. SE-12, No. 7, pp 744-751, July 1986.

Gilb, T. and Graham, D., *Software Inspection*, Addison Wesley, 1993.

Gilb, T., *Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*, Elsevier Butterworth-Heinemann, 2005. ISBN: 0750665076.

Haley, T., B. Ireland, Ed. Wojtaszek, D. Nash, R. Dion. Raytheon. 1995. Raytheon Electronic Systems experience in Software Process Improvement. This paper is available on-line at http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.017.html

Humphrey, W. S., *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.

The author is extremely grateful for editing assistance from Lindsey Brodie, Middlesex University, UK, on this paper.

Biography

Tom Gilb is the author of 'Competitive Engineering: A Handbook for Systems & Software Engineering Management using Planguage' (published in June 2005), 'Principles of Software Engineering Management' (1988) and 'Software Inspection' (1993). His book "Software Metrics" (1976) coined the term and was used as the basis for the Software Engineering Institute Capability Maturity Model Level Four (SEI CMM Level 4). His most recent interests are development of true software engineering and systems engineering methods.
Tom Gilb was born in Pasadena CA in 1940. He moved to England in 1956, and then two years later he joined IBM in Norway. Since 1963, he has been an independent consultant and author. He is a member of INCOSE.

The Reality of Software Testing in an Agile Environment.

Ten QA Myths Blown Apart

by George Wilson

The definition of agile testing can be described as follows: "Testing practice for projects using agile technologies, treating development as the customer of testing and emphasizing a test-first design philosophy. In agile development, testing is integrated throughout the lifecycle, testing the software throughout its development."*

Agile is a methodology that is seeing increasingly widespread adoption and it is easy to understand why—especially if you consider the developer and user point of view.

Users: Don't want to spend ages being quizzed in detail about the exact requirements and processes for the whole system, and then have to review a large specification, which they know could come back to haunt them.

Developers: Don't want to have to follow a tight specification, without any expression of their own imagination and creative talents, especially if they can see a better way.

Yet for the QA professional an Agile approach can cause discomfort – In the ideal world they would have a 'finished' product to verify against a finished specification. To be asked to validate a moving target against a changing backdrop is counter intuitive. It means that the use of technology and automation are much more difficult, and it requires a new approach to testing, in the same way that it does for the users and the developers.

All the agile approaches have (at least) one characteristic in common in that they impact the role of the QA professional. This in itself is not a bad thing when the outcome is a step change for the better. However, when decisions are made on the basis of an invalid paradigm, change is not always analogous with progress. When a new paradigm is proposed for software development, by software developers, it is not a surprise that it is developer-centric. Abraham Maslow said that "*He that is good with a hammer tends to think everything is a nail.*" The responsibility of the QA profession is not to bury its head and pretend that agile development will go away, it is our responsibility to engage in discussion to ensure that someone with a hammer is not pounding on a screw!

With the emergence of Test Driven Development** some suggest the role of QA is now questionable citing Test Driven Development (TDD) as the key to testing. But, what is most important is that QA is directly involved in the agile scrums all the way through, to be an integral part of the team designing the tests, at the same time as the requirements and solutions evolve.

QA teams need to know the real impact of an Agile methodology, there are boundless myths circulating the industry. Here is our reply to ten of our favorite myths:

Ten QA Myths Regarding Agile Testing.

There are a number of comments and statements regarding TDD and the QA function beginning to appear in articles on the internet by so-called specialists, which are, at best, misguided. This article responds to some of these myths and highlights challenges that QA teams will need to face up to and address in order to be successful in an increasingly agile world.

Myth One: "You only need to unit test - TDD testing is sufficient"

For the vast majority of commercial developments this simply isn't true. Even strong proponents of agile development recognize the need for their armory to include a range of testing techniques. For example, Scott W. Ambler has a list of 21 different testing techniques as part of his FLOOT (Full Life Cycle Object-Oriented Testing) methodology, including white box, black box, regression testing, stress testing and UAT. (http://www.ambysoft.com/essays/floot.html)

TDD programmers rely on these tests to verify their code is correct. If the requirements (test cases) are specified incorrectly, then you'll build robust code that fails to meet the objective.

Therefore, most agile projects include investigative testing efforts (black-box) which support rather than compete with white-box testing. Good investigative testing will reveal problems that the developer missed before they get too far downstream.

Myth Two: "You can reuse unit tests to build a regression test suite"

Some TDD proponents argue that conventional downstream testing is unnecessary because every line of application code has a corresponding test case; they suggest that by reassembling unit tests, everything from User Acceptance Tests to Regression Tests can be performed.

Although this sounds feasible, it isn't realistic, and here's why:

The granularity and objectives of white-box unit tests developed in TDD serve a different purpose from downstream black-box testing.

While the overall objective of a unit test is designed to prove that the code will do what is expected, the aim of regression testing is to ensure that no unexpected effects result from the application code being changed. These two

objectives are not synonymous – e.g. checking that an attribute has a valid date format, is not the same as checking that for a given input, the value of the field contains an expected date.

Myth Three: "We no longer need testers, or automation tools"

Professional testers fulfill a different and equally valid role from their development colleagues.

It is widely recognized that traditional test automation tools have failed to live up to the hype of the vendors. Original Software's roots are as providers of products that improve the productivity of the database testing environment. It was because there were no adequate tools to provide User interface testing that we extended our solutions into this market sector; our heritage is aligned to effective testing rather than screen-scraping automation. When we developed TestDrive, we did it with the benefit of twenty years hindsight of missed opportunities from the traditional vendors.

Often, TDD projects have at least as much test code as application code and, therefore, are themselves software applications. This test code needs to be maintained for the life of the target application.

Myth Four: "Unit tests remove the need for manual testing"

Manual testing is a repetitive task; it's expensive, boring and error-prone. While TDD can reduce the amount of manual effort needed for functional testing; it does not remove the need for further black-box testing (manual and automated).

By automatically capturing the tester's process and documenting their keystrokes and mouse-clicks, the tester will have more time for the interesting, value-add activities, such as testing complex scenarios that would be difficult or impossible to justify automating. Though manual testing is a time-consuming (and therefore expensive) way to find errors, the costs of not finding them are often much higher.

Myth Five: "User Acceptance Testing is no longer necessary"

Within agile development, acceptance testing is often positioned as working with the customer to resolve "incorrect requirements", rather than correcting functionality that does not map to what the user needs. When the user initially defines their requirements, they do it based on their initial expectations. When they see the system "in the flesh" they will invariably come up with different, or additional, requirements. While agile methods might reduce the frequency of this happening, it is unreasonable to expect the approach to resolve them entirely, so there should be no expectation that user acceptance testing will be obviated. This is especially true when it comes to the business user signing off approval of the User Interface, since they may have envisaged something different from the developer; which brings us nicely to myth six...

Myth Six: "Automation is impossible"

Automation in the early stages of an agile project is usually very tough, but as the system grows and evolves, some aspects settle and it becomes appropriate to deploy automation – automation that can cope with changes by using approaches such as self healing scripts.

To begin with, almost all testing from a user and QA perspective will be manual, but this testing effort and design can be beneficial later if captured and re-used.

Knowing the right time to automate is tough, so using technology that proactively supports early manual testing, but also provides a path to evolve this into automation, is the key.

Myth Seven: "Developers have adequate testing skills"

If testing was easy, everybody would do it and we'd deliver perfect code every time.

An independent testing team serves as an objective third-party, able to see "the big picture", to validate the functionality and quality of the deliverable. While developers tend towards proving the system functions as required, a good tester will be detached enough to ask "what happens if…?" When you include business user testing as well, you are more likely to have a system that is fit for purpose.

Finally, and I'm sure this point will be disputed, most developers don't actually want to spend much time first writing tests and then developing code to prove the tests work. Using the collaborative process described below, the developer gets ample assistance in describing the functional tests and can focus on writing lean, accurate and robust code.

Myth Eight: "The unit tests form 100% of your design specification"

Whichever development method you use, before you develop code you have to think about the requirements. While TDD "done well" may identify a fair percentage of the design specification, there are still concerns about gaps between the unit tests. There are other equally viable approaches. The argument, that you need TDD to prove the requirements are captured accurately, isn't substantiated by history.

Defining test cases to check the accuracy and succinctness of the requirements is nothing new. The V-model, for example, is a valid approach to understanding testing requirements – and by implication, functional requirements. Like TDD, the V-model and most other models, fall down when the practitioner's approach is rigid, while development processes are fluid. Software engineering is not like mechanical engineering and trying to force conformity is wasted effort. Whichever approach you chose, correct thinking challenges each user requirement by asking "how would I test that?" The important factor here is that test cases are challenged before they are committed to code, otherwise you'll be doing a lot of recoding and calling it "refactoring".

When the requirements are refined through collaboration, the developer receives a more robust specification that is less likely to change because it has been openly appraised from several people's perspectives.

Myth Nine: "TDD is applicable on every project"

As the size of the project increases, the time required to run the tests also increases. This can be addressed by partitioning either/both the project and/or the tests. Either route results in tests that run at different frequencies depending upon their relevance to the current coding effort. This approach introduces the need for test planning and execution management. To achieve this efficiently, in addition to white-box testing, you need to be thinking about:

Integration Testing – "Which tests do I need to run to ensure the new code works seamlessly with the surrounding code?"

System Testing – "Does the functionality supported by the new code dovetail with functionality elsewhere in this system, or in other systems within the process flow?"

Regression testing – "How often do I need to run a regression test to ensure there are no unforeseen impacts of the new code?" Automated regression testing provides a safety net that can affirm agile development techniques.

User Acceptance Testing – "While TDD (in collaboration with business users) should ensure that a specific function performs correctly, is the cumulative impact of changes still acceptable to the business users?"

However, in today's environment it is unacceptable to think of these testing stages as discrete serial activities. Often they will need to be run in parallel as we get a new 'code drop'. As the size of the project team increases (along with testing effort), it is no longer acceptable for the tests to be considered "self-documenting". Whenever the number of participants increases, the project's risks become exposed to its members' different interpretations of the requirements – the definition of what constitutes "success" can be misconstrued.

As the size of the project increases, so would the amount of test code that needs to be developed. Any test code developed needs to be supported for the life of the application – effectively doubling the ongoing maintenance effort.

As the size of the testing workload increases the project needs to add test automation to its armory, to minimize the human intervention and elapsed times for each of these testing cycles.

Myth Ten: "Developers and Testers are like oil and water"

Since the dawn of time there has often been a "them and us" tension between developers and testers. This is usually a healthy symbiotic relationship which, when working correctly, provides a mutually beneficial relationship between the two groups resulting in a higher quality deliverable for the customer.

The discussion should be focused on:

- Software delivery that meets business objectives (fit for purpose, on time and on budget), not who owns which part of the process.
- What can testers contribute to the requirements gathering phase to ensure they are involved in the TDD process?
- How can testers maximize reuse of the assets created during the development phase?
- Is there a role for the "traditional tester" in TDD, or should they (like the developers) be acquiring new skills to enable them to adapt to the new paradigm?
- How can developers and testers find mutual benefit in exploiting new advances in software development and testing tools?

Just as improvements in developer's software tools and methods have enabled a shift in development approaches, next generation technology for test automation is similarly reframing the opportunities for testers to automate earlier in the delivery cycle without incurring the heavy burden of script maintenance so often associated with traditional automation tools.

Conclusion

Agile projects are in fact an excellent opportunity for QA to take leadership of the agile processes; who else is better placed to bridge the gap between users and developers, understand both what is required, how it can be achieved and how it can be assured prior to deployment? QA should have a vested interest in both "the how" and "the result", as well as continuing to ensure that the whole evolving system meets the business objectives and is fit for purpose. But it requires QA professionals to be fluid and agile themselves, discarding previous paradigms and focusing on techniques to optimize a new strategy to testing.

* Definition taken from The Glossary of Software Testing Terms from Original Software (www.origsoft.com)

**Defined on Wikipedia as "A software development technique consisting of short iterations where new test cases covering the desired improvement or new functionality are written first, then the production code necessary to pass the tests is implemented, and finally the software is refactored to accommodate changes."

© iStockphoto.com/ Andresr

Biography

George Wilson, Founder and Operations Director of Original Software

George is Operations Director responsible for sales, consulting services & product support delivery to Original Software's worldwide client base.

Another founding member; George's strong software quality and customer care orientation is reinforced by extensive software product and large-scale development project experience within many areas of IT.

He has helped shaped the solutions Original offer today with practical experience from the field, combining well with Colin's insight and innovation. An engineer by training, his background served him to great effect at Osprey Computer Services (to 1995) where, as a main board director, he drove development and marketing of new applications into new markets for the company.

Later, as Business Group Manager at AIG Computer Services, George rapidly broadened his platform experience, simultaneously managing IBM Midrange, NT and PC development projects - in a rigorous ISO9001 and TickIT management environment, where George's natural 'quality evangelism' served him well.

He has been a keen dinghy sailor, enthusiastic windsurfer but these days golf seems to take precedence.

Years of experience in providing professional services and in building R&D centers and testing labs all over the world, helped us to solve the equation and to be able to offer you end to end solutions of high quality, still using a competitive pricing scheme.

Some key facts about SELA:

Microsoft Gold Certified Partner	Accredited Training Provider - ISTQB
Leading the deployment of the new Windows 7 around the world	800 man-years experience in consulting and R&D activities
18-year track record	Locations: Canada, USA, Israel, India and Singapore

Worldwide professionals in the following fields and cutting-edge technologies:

Software Testing	**.NET**	**J2EE**	**Embedded Real-Time**
Application Security	**VSTS**	**C++**	**Agile and Scrum**

SELA can provide you services around the globe.
Contact us, to start solving your equations.

s o l u t i o n s @ s e l a . c o . i l
w w w . s e l a . c o . i l / e n

Microsoft

SELA TECHNOLOGY CENTER

ISTQB

Beyond Functional Testing: On to Conformance and Interoperability

by Derk-Jan de Grood & David Bakker

Introduction

Conformance testing and interoperability testing are less well-known than functional testing, performance testing or usability testing. Moreover, they may seem dull to some people. They are all about standards, thick technical specifications, and repetition. Nevertheless, Derk-Jan de Grood and his colleagues believe that conformance and interoperability testing are as important as other types of testing and can be interesting, challenging, and rewarding. This article highlights important concepts such as interface standards, multi-vendor architectures and oracles. It provides insight into the problems you might have to deal with when you move on to conformance testing.

Why Bother About Interoperability?

Imagine that you have a business process that you want to automate. Once the newly-built system is implemented, the automated business process should replace the original one. The new system is specially designed for your unique situation. The system probably has some interfaces with other systems, but these are all trusted and known. While designing and building such a system, the 'fit for purpose' will be a leading requirement. Testing should provide the answer whether the system is actually 'fit for purpose'. In your test process, validating that the systems works will be more important than the question how it works.

There are, however, many real-life systems that do not have a one-to-one relation with trusted peer systems. Instead, they operate in an open environment and communicate with a lot of different systems built by various manufacturers. In these kinds of systems, interoperability between systems of different vendors and owners is a key issue.

Think about bank cards, ID-badges and electronic passports. In all of these systems we recognize an owner that should be able to use his/her card with a wide variety of different machines. In the case of a bank card, these can be ATMs from different manufacturers and banks. An electronic passport will be read by different types of passport scanners in various countries all over the world.

Interoperability is also important within service-oriented architectures (SOA). Within an SOA, the system often calls services that are outside the boundaries of the organization. A typical example is that of a web shop that helps customers find the cheapest offer. The web shop requests availability and price information at various shops that have an SOA interface. Again, we recognize a one-to-many relationship.

A third example where interoperability plays a major role is embedded software. Manufacturers of consumer electronics reduce their time-to-market by integrating a growing number of commercial-off-the-shelf (COTS) components into their products. Each of the components should be compatible with the main system and also be able to communicate with other components within the device. But it's important to remember that the component is probably used in other devices by other manufacturers as well. Once more, we recognize a one-to-many relationship.

In order to make these open environments work, agreements are made about the working of the system and the way data is exchanged with peer systems. By complying with those specifications or standards, the interoperability of the systems is guaranteed.

If we decide to build according to a standard, we should also test whether this has been done properly. This is where conformance testing and interoperability testing kicks in.

Conformance and Interoperability

Intuitively, we all know what is meant by conformance and interoperability. Yet it might be useful to define these terms more precisely.

Conformance is the measure to which a system meets the requirements of a standard.

Compliancy is often used as synonym for conformance.

Interoperability is the ability of different systems to work together effectively and to exchange information. We distinguish horizontal and vertical interoperability:

- *Vertical interoperability* refers to different types of components that work together in a chain. In the example of the electronic passport, the chain can consist of the passport, a reader, an on-site server and a back-office system. Testing vertical interoperability aims, for instance, to prove that it is possible to match biometric data on the chip of a passport with data stored in a database in the back-office system. Vertical interoperability is very important to the different stakeholders in the chain, such as the passport holder and the receiving country.
- *Horizontal interoperability* refers to the exchangeability of components of the same type. Again, looking at the e-passport example, there are different countries that issue electronic passports, and there are multiple manufacturers that deliver e-passport readers. Testing horizontal interoperability checks whether the working of the chain is independent of the components used. Horizontal interop-

erability is a key asset in multi-vendor architectures.

Multi-vendor architectures

Conformance and interoperability play a major role in multi-vendor or 'open' architectures. These are architectures where more than one manufacturer (vendor) is allowed to deliver parts for the system. Advantages of multi-vendor architectures are that they prevent vendor lock-in and promote competition. The system owner is assured of his independence towards the manufacturers. He will not have a problem if one of the suppliers is unable to deliver. This independence leads to healthy competition and stimulates innovation.

Interoperability and Conformance Testing

Testing for interoperability can be done in different ways. The most straightforward way is by performing a *cross-over test*, in which all combinations of different systems are tested. Sometimes this type of testing is also referred to as *matrix testing*, indicating that each position in the interoperability matrix should be tested. The example below depicts a 2-dimensional matrix for different kinds of ATM-cards in different cash machines.

Card type works with ATM type?	ATM 1	ATM 2	ATM 3	ATM 4
Card I.a	√	√	√	√
Card I.b	√	√	√	x
Card II	√	√	√	√
Card III	√	x	√	√

Example of an interoperability matrix

Cross-over testing has three disadvantages:

- If the number of different systems increases, the test√ing effort and time will explode to unrealistic measures.
- It is not possible to include systems within the test that have not yet been delivered. How can one ensure the interoperability of your bankcard with future systems, such as an ATM that will be on the market next year?
- Problems are generally found quite late in the development cycle: only systems that are considered ready are put to a cross-over test. The efforts to find the cause of the problem and to repair it will therefore be relatively high. It is not unlikely that problems that are found in one particular combination will lead to a change in the working of the interface. In this case, a lot of re-testing is required in order to determine that the change does not affect the working of the other combinations.

Since cross-over testing is difficult and time-consuming, one prefers to execute, prior to integration, a run of *conformance tests* on each of the system interfaces.

Conformance tests capture the technical description of a specification and measure whether a product faithfully implements the specification. The testing provides developers, users and purchasers with a higher level of confidence in product quality and increases the probability of successful interoperability [adapted from NIST].

The basic assumption behind conformance tests is that if two systems work according to the same standard, they can integrate or be exchanged with one another without problem. Passing the tests in the conformance test set will therefore reduce the number of problems found when two or more systems are integrated (vertical integration) and will lead to exchangeable components (horizontal integration).

Since the proof of the pudding is in the eating, the conformance test is often followed by an *End-to-End* or *Chain* test. In such a test, all components are put together to see whether the system as a whole functions correctly. The End-to-End tests are not as extensive as the 'cross-over' test. Rather than testing each possible combination, it checks whether the whole chain is working for a representative selection of the available components.

Finally, upon integration, a *check- to-connect* can be carried out. The check-to-connect test checks whether the component is connected and configured correctly.

Drawing up a test design for conformance tests

When designing tests for conformance testing, the tester will have to deal with the following challenges:

Incomplete and ambiguous standards

A regularly recurring experience in any software development project is the lack of complete and consistent specifications. Although interface standards describe limited functionality, they also fail to be complete at the beginning. The reason for this lies in the difficulties of writing complete specifications and in the way standards are created. A standard is often a compromise between different parties with different interests. For example, a group of competing manufacturers and operators in the area of public transport may decide to cooperate on a standard. By creating an interoperable system they will open up new markets and create new chances. They will have a number of requirements in common, and these will find their way into the standard quite easily. But each of the manufacturers will also have some features in their existing products that distinguish them from their competitors' systems. These points will lead to a lot of discussion among manufacturers. Often, the manufacturers will settle for a 'high-level' solution, in order to stop the discussions and get the standard finished in time.

Testing is of course all about the details, and more than once the tester's questions will trigger the discussion once more. This can lead to a delay in the project. Often this problem is mitigated by creating an '*oracle*'. An oracle is able to make straight and firm decisions that are followed by all parties. This can be seen in the EMV scheme. The success of EMV lies partly in the fact that VISA and MasterCard are such strong parties. When VISA and MasterCard agree on a solution, all other parties will follow. For this reason, VISA, MasterCard and JCB operate EMVCo, which acts like an oracle that decides upon all issues regarding EMV and prevents endless discussion by the members.

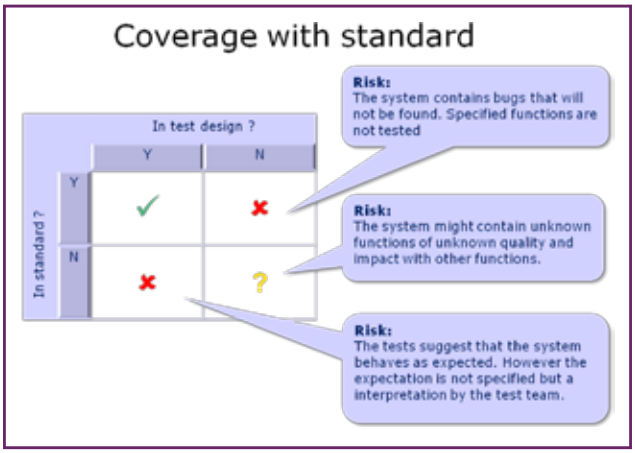
Incomplete test set

Conformance tests are carried out to minimize the number of problems that arise when two or more systems are integrated. The test should lead to a 'declaration of conformance'. Conformance testing is therefore all about coverage of the standard.

Experience shows that designing tests with full coverage is quite a challenge.

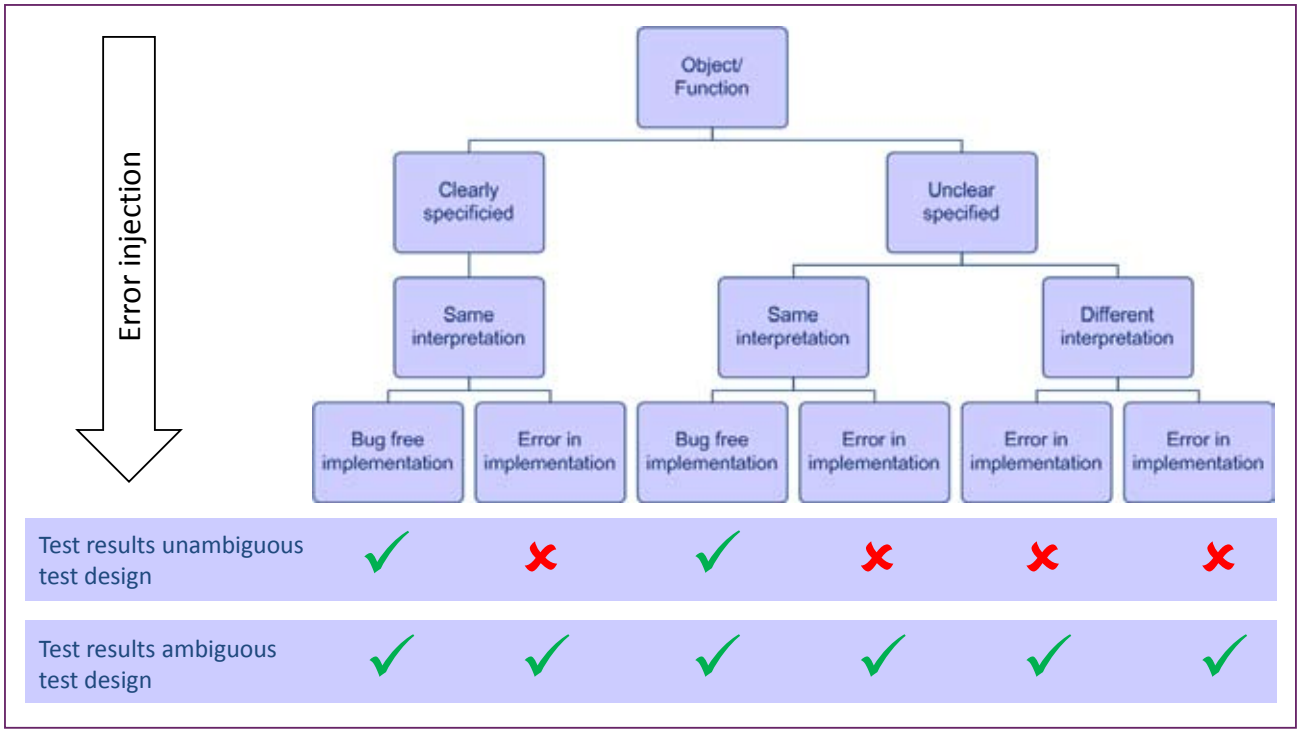
The international e-passport is specified by the International Civil Aviation Organization (ICAO). ICAO also publishes a test standard for both e-passports and e-passport readers. An assessment that Collis did on these test specifications found that they did not cover all requirements. In particular, the test set did not contain enough negative testing, and tests for the correct formatting of the biometric data in the passport were missing.

The figure below illustrates the relation between standard and test design. Risks are indicated if there are omissions in either standard or test design.



Ambiguity of the tests

If you leave room for interpretation in the test set or standard, different manufacturers will



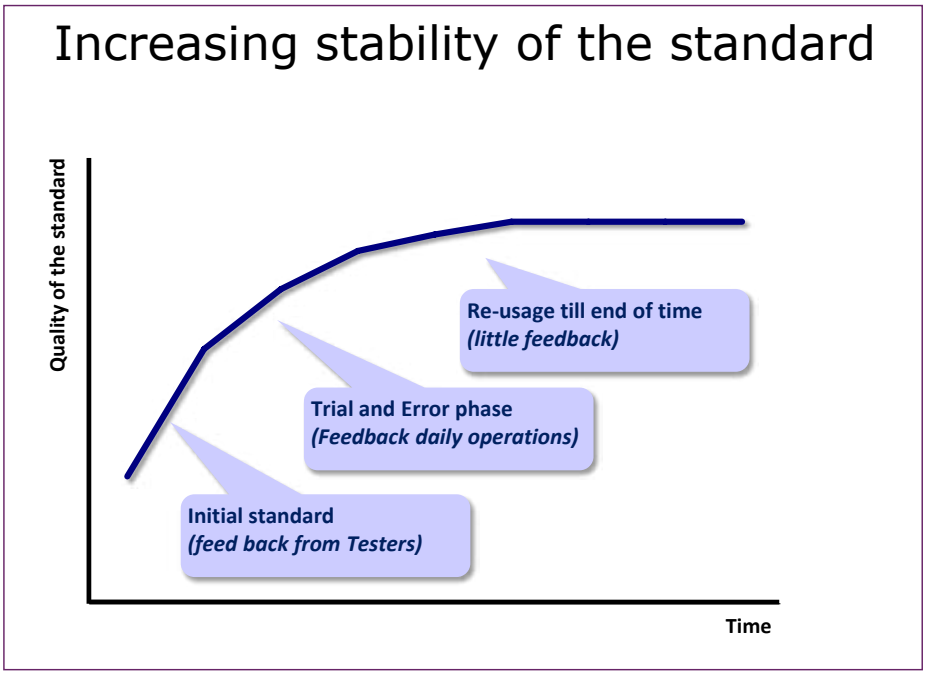
deliver different implementations. What does the outcome of the test really say about the system's behavior? An ambiguous test might lead to a positive verdict on systems with different behavior, jeopardizing interoperability between those systems.

The importance of an unambiguous test set is illustrated in the figure below. The figure indicates how errors can sneak into a certain function. Before the developer implements the function, he will make an interpretation of the design. When the function is unclearly specified, the chances are quite high that he will make a different interpretation of the function than the analyst had intended. When implementing the function, the developer can either make an error, or not. The figure below shows that in four of the six situations the conformance tests should indicate a deviation from the expected behavior.

A clear and unambiguous test will filter out these erroneous situations and prevent that two different implementations pass the same test. An unclear test will not. Therefore it is a challenge for the tester to write tests in such a way that the result cannot be disputed.

Conformance testing: a guarantee for interoperability?

Many say that interoperability through conformance testing is a myth. And indeed, practice shows that interoperability problems show up even for systems for which outstanding conformance tests exist. The credit card issuer VISA relies on trustworthy and error-free services all around the world. In order to enable this, VISA is one of the driving forces behind the EMV standard and the accompanying test standard. Although EVMCo invests heavily in good standards and reliable conformance tests, VISA still runs a dedicated department dealing with interoperability issues. Although conformance testing has decreased the number of problems to a great extent, remaining errors are still found in production.



Myth or not, by gathering the results from subsequent tests and from daily operations, the EMV standard gets updated with renewed insights and the conformance test set has been fine-tuned over the years. Each credit card, ATM or back-office server that has been added to the system has increased the quality of the EMV standard and of the conformance test set. And in turn, the increasing quality of the standards leads to less and less problems when new equipment is added to the system. Therefore, the quality of the standards is nearing the 'perfect' mark asymptotically, as shown in the figure below.

This is where we find the true added value of conformance testing: these tests save a lot of problems during integrating and can be reused over a long period of time for many different new components.

Biography

Derk-Jan de Grood has broad, hands-on experience as test engineer, test manager and consultant in various industries. As manager of a test department he learned how to implement test methods the practical way. He gives lectures at various Dutch universities and is author of the first educational book specially written for teaching software testing at Dutch universities. Derk-Jan is also the author of TestGoal, the result-driven test philosophy by Collis. As an ISTQB/ISEB practitioner certified test manager, he coaches the test experts of Collis. Besides that, he is a passionate, inspiring speaker at major testing conferences all over the world such as the STAR conferences in the USA and Europe.

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

160.000 readers

ADVERTISEMENTS
sales@testingexperience.com

SUBSCRIBE
www.testingexperience.com/subscribe.php

PRICE
digital version: free of charge
print version: 8,00 €

Díaz Hilterscheid

ISSN 1866-5705

Yaron's Forecourt

Outsource your testing – Is it really worth it? Some tips...

by Yaron Tsubery

Dear colleagues and readers,

I hope that the winter didn't catch you unready and you had a great time with your families and friends over the holidays. Welcome back and please feel free to join me for a glass of chilled wine or what ever you choose… so, are you ready?…

Outsource your testing… Hmm, well, I would like to start by sharing with you my first adventure with outsourcing: when I was a junior test engineer and looked for a job I saw that one of my options was to join a leading test consultant company as one of their employees. At that time I had many other options too, so the debate was whether to choose a consultant company who were experts in testing, to learn from experts, and participate in interesting and challenging projects (I hoped…) or to join a high-tech company and hope that I would get the same. In the end I chose to join a high-tech company. A long time has passed since then, and now-a-days, as a manager, I see things a bit differently. However, before continuing, just remember that this is your choice to decide – where to be and with whom to work.

My peers (managers like myself) and I ask ourselves about this a lot - isn't it better to have your own personnel near you and employed by your company? Well, probably yes…but the reality is totally different. We as managers have to follow the company's rules and procedures; we already know that salaries are one of the biggest items on a company's budget, and hiring is part of the strategic plans. A short story if I may: one of the managers I know told us that she had a team who worked under an off-shore contract, and they didn't perform well. She approached her manager asking him to stop the contract and hire sub-contract employees to work in the company – she even suggested reducing the number of employees, claiming that it would be much easier for her to manage a small number locally than a larger group off-shore, and she was sure that they would perform better. Her manager looked at her and said: "…you know, you have no choice, manage them or lose them, because the company will not hire subcontractors now, if they have an outsourced team…". "What to do, Yaron?" she asked; "Well" I told her "I have a team in India, and together with the team leader, the team members there, and a contact person in Tel-Aviv we did a small revolution – we created a continuous improvement plan, which included all procedural, professional and product/system knowledge items. We held sessions of knowledge transfer, aligning expectations and devised ways to improve the communication between the teams, our contact person took a special seminar on Indian culture and he shared with them some aspects of Israeli culture too. We held a monthly review on the performance and improvements of the team, and now the team is stable, performing well and progressively improving." So you see, the skies are not so high, you just need to be focused, define the way forward and recruit both sides to your mission. "A few months ago…" I told her "we started to work on the next stage, which is: Team Independence". "So, believe that you can do it too" I said, "get advice from your colleagues on how they succeeded with out-sourcing, and finally you'll find yourself dealing with it rather than neglecting it…" (We, as managers, are not expected to neglect things, we, as experts and senior managers, are expected to deal with the situation and find solutions!)

What about "fixed price" or lets say project based employees? Well, this is a totally different story. This option is mainly for a known project scope that you can combine in a so called "package" and give someone else to do – it is often the case when you don't have the domain knowledge. However, if you think that you can give that part of the project to someone else, and then come back at the end and it will all have been done, then you're wrong. You must manage it as a project like any other, with information and updates on progress, stability and reliability of the team. I'll give you an example from a previous project we had, where we were required to perform security tests for the system. After sitting down together with the project manager and the system architect, I decided to hire a company who are experts in security testing and outsource this part of the testing to them. In the beginning, the project manager was afraid that he would not get as good quality as he gets from my teams, but after I explained to him that this is not my domain, I'm not an expert in it, and it would be better to hire a specialist company for that, he agreed and we went ahead, (of course, only after performing due-diligence on the company, and signing an SOW (Statement Of Work) that describes the scope of work and targets). At the end it was a wise choice that saved us time.

"Great", you are probably saying, "but what about cases that don't have a good ending?" To that I reply: "You are right and I've another story for you… but beforehand remember, most of the time we learn from our mistakes, but it's much better to learn from others'". I promised a story: "…well, having a large group of test engineers, in which most of the employees are not "yours" is a bit dangerous - you need to take care not to lose them, you need to educate, coach, train and sharpen their experience and skills. I lost two employees that I cared about a lot, because I didn't define a clear career roadmap for them, and get their commitment and agreement on it. Afterwards, I felt that something needed to be done - I learnt my lesson and now I know that it's not enough to have this plan, you need to align it with the employee, and you must get approval from your company, your manager and the company that employs him or her… I learned that I'm not the only stakeholder in this process. You could say that the same goes for your own employees and you'd be right. However, the difference is that our nature lends itself to caring for the people closest for us, in this case, our own staff. It's obvious that you will look after them and their needs. It takes much more effort to act the same way towards your "fixed price" staff, since they are not really your corporate responsibility.

In short my friends, outsourcing is a fact of life, and we can choose to join it, we can choose to manage it, and we can choose to work with it. There is more to learn about it – not from books but from articles, experiences and shared information (you might add conferences to your list too).

There's a storm going on outside, with thunder and lightening, and it is reminding me that its late and we need to end. I hope that you got some tips for the road and you know that you're not alone… I hope that you enjoyed and you're always welcome to ask questions on anything that you wish to share, don't be shy – send them to the following account, prepared especially for our discussions: yaron@testingexperience.com

Yours truly, Yaron

Biography

Yaron Tsubery has been a Director of QA & Testing Manager at Comverse since 2000. Yaron is in charge of a large group that includes Testing Team Leaders and Test Engineers.
Yaron has been working in software since 1990, and has more then 17 years in testing field as a Test Engineer, Testing TL, and Testing Manager.
Yaron is a member of ISTQB (International Testing Qualifications Board – www.istqb.org) and is the President and founder of the ITCB (Israeli Testing Certification Board – www.itcb.org.il). He is a member of IQAMF (Israeli QA Managers Forum) and in SIGiST Israel.
Yaron has been invited as a speaker to some important international conferences to lecture on subjects related to testing.

Training with a View

Díaz Hilterscheid

also onsite training worldwide in German, English, Spanish, French at
http://training.diazhilterscheid.com/ training@diazhilterscheid.com

Kurfürstendamm, Berlin © Katrin Schülke

23.03.09-26.03.09	Certified Tester Advanced Level - TEST ANALYST	German	Düsseldorf/Köln
30.03.09-01.04.09	IREB - Certified Professional for Requirements Engineering - Foundation Level	German	Berlin
06.04.09-09.04.09	Certified Tester Advanced Level - TESTMANAGER	German	Düsseldorf
20.04.09-22.04.09	Certified Tester Foundation Level	German	Berlin
21.04.09-23.04.09	**Certified Tester Foundation Level**	**French**	**Paris**
27.04.09-29.04.09	Certified Tester Foundation Level	German	Frankfurt
27.04.09-30.04.09	Certified Tester Advanced Level - TECHNICAL TEST ANALYST	German	Berlin
04.05.09-07.05.09	Certified Tester Advanced Level - TESTMANAGER	German	Berlin
11.05.09-13.05.09	Certified Tester Foundation Level	German	Hannover
11.05.09-14.05.09	Certified Tester Advanced Level - TEST ANALYST	German	Berlin
18.05.09-20.05.09	Certified Tester Foundation Level	German	Berlin
19.05.09-21.05.09	**Certified Tester Foundation Level**	**French**	**Paris**
20.05.09-22.05.09	**Certified Tester Foundation Level**	**English**	**Palma de Mallorca**
25.05.09-27.05.09	IREB - Certified Professional for Requirements Engineering - Foundation Level	German	Berlin
08.06.09-10.06.09	Certified Tester Foundation Level	German	Berlin
15.06.09-18.06.09	Certified Tester Advanced Level - TESTMANAGER	German	Berlin
18.06.09-20.06.09	**Certified Tester Foundation Level**	**French**	**Paris**
22.06.09-24.06.09	Certified Tester Foundation Level	German	Munich
29.06.09-02.07.09	Certified Tester Advanced Level - TESTMANAGER	German	Frankfurt
01.07.09-03.07.09	**Certified Tester Foundation Level**	**French**	**Paris**

"Thanks for the entertaining introduction to a complex topic and the thorough preparation for the certification. Who would have thought that ravens and cockroaches can be so important in software testing"
Gerlinde Suling, Siemens AG

"Mr. Lieblang's great wealth of experience, his excellent lecture style and his refreshing manner made the 3-day ISTQB training into an experience. With this preparation passing the subsequent examination was child's play - which our course result of 94.5% clearly proves."
Stefanie Schlegel, T-Systems Multimedia Solutions GmbH

"A casual lecture style by Mr. Lieblang, and dry, incisive comments in-between. My attention was correspondingly high. With this preparation the exam was easy."
Mirko Gossler, T-Systems Multimedia Solutions GmbH

"Thank you for 3 very informative days. I passed my exam with 39 out of 40 questions correct and came away with new inspiration for my work."
Rita Kohl, PC-Ware Information Technologies AG