December 2012

testing experience

The Magazine for Professional Testers

Testing in the Cloud

printed in Germany
print version 8,00 €
free digital version
www.testingexperience.com
ISSN 1866-5705

AGILE DEV PRACTICES

The Conference for Agile Developers

Retrospectives on Agile - Forward Reasoning

March 4-7, 2013
in Potsdam/Berlin, Germany

Choose among eight full-day tutorials with:

Olav Maassen

Andrea Tomasini

Carlos Blé

Bob Galen

Peter Saddington

Enrique Comba Riepenhausen

Ellen Gottesdiener

Pawel Brodzinski

The early bird swings its wings until January 31, 2013 - try to catch it and save up to 400,00 €.

www.agiledevpractices.com

Sponsors & Exhibitors 2013

codecentric | iSQI | testing experience | Agile Record | Díaz Hilterscheid

Dear readers, in a few days it will be Xmas and you will get some gifts from Santa Claus or “los reyes magos”.

We got already some gifts this year. A little bit earlier, but I think we deserved them. We did have the best conference ever. Speakers, attendees, sponsors and organizers were very happy about the success of the **Agile Testing Days**.

It was amazing to see how people interacted, and exchanged knowledge and experiences. Many of them found new friends in the community and developed deeper relationships in some cases.

We are already looking forward to planning the new one. Once again, I thank you all for your support, for attending the conference, and for giving us very good marks. Last but not least, congratulations to Lisa Crispin. She got the **MIATPP Award** this year.

We are now looking at **Agile Development Practices**. A new conference. Based on our experience, it looks like it is going to be a great one, too. Please tell your developers about it.

For the first time, AQIS is running the **Belgium Testing Days** on its own. Have you seen the program? Please take a look. Mieke Gevers did another great job there and introduced a lot of new things that will help to keep the conference on this successful track. Please ask your contacts who may be interested. It is not only the best testing conference in Belgium, it is also a reference in Europe for all professional testers.

In this issue you will find a lot of articles on **cloud testing**. Once again we have a high number of them. Thanks to all the authors and supporters for this new issue.

We have developed some new things for next year. These will be our gifts to you. Stay tuned.

We wish you a Merry Xmas and a Happy New Year!

Yours sincerely,

José Díaz, Editor

contents 20/2012

Moving Testing to the Cloud – An Exploration

Cloud computing recently received significant attention because it changes the way computing and services are delivered to customers. For example, it changes the way computing resources, such as CPUs, databases, and storage systems are provided and managed. Today, leading players such as Amazon, Google, IBM, Microsoft, and Salesforce.com are offering their cloud infrastructure for services.

27

Software Testing: Adapt and Grow with the Cloud

Cloud computing has certainly brought a lot of new possibilities in terms of business and, clearly, as the arrival of a new technology or methodology often requires, software testing needs to adapt and grow in this direction.

19

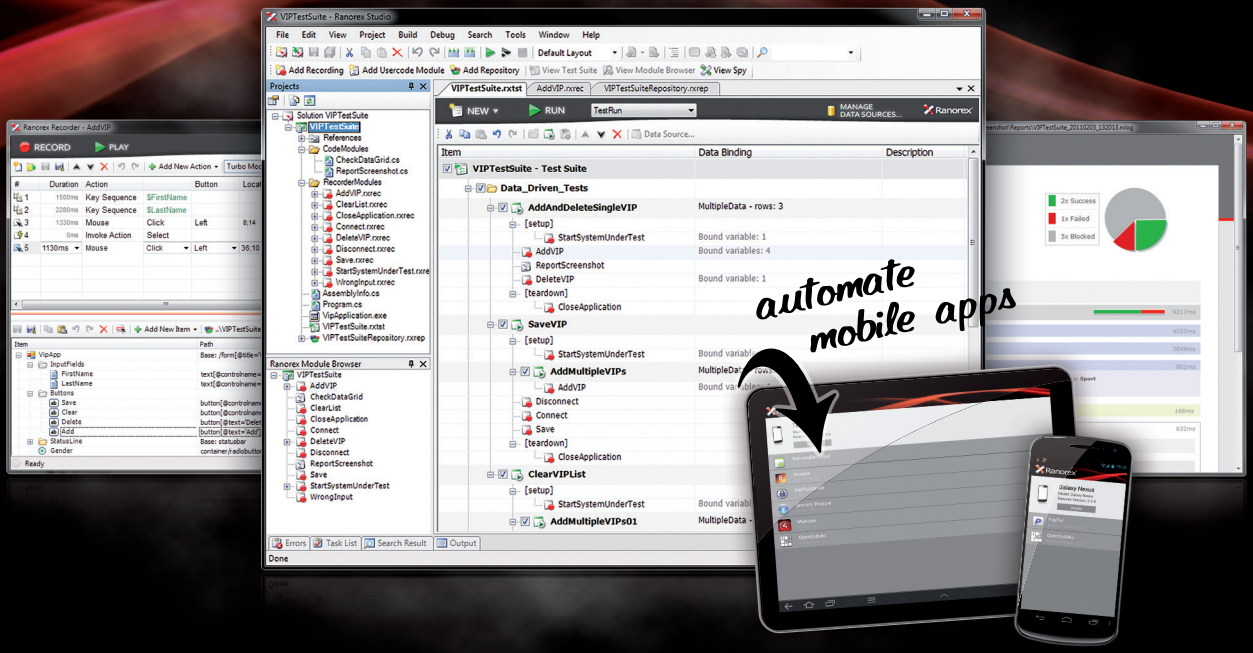
Indentifying and Fixing Requirement Defects in Later Stages Increases Costs

The core focus of this paper is to inform and enhance the view of software process management and metrics, and dynamic process modeling.

43

Ranorex®

Automate Testing of any Windows, Web & Mobile App



- ✓ Use connectors for data-driven tests
- ✓ Build robust test automation frameworks
- ✓ Generate EXEs for pure flexibility
- ✓ Write custom code in C# or VB.NET

Record and Edit Reliable Test Actions

Manage and Execute Your Automated Tests

Reproduce Bugs and Maintain Your Tests

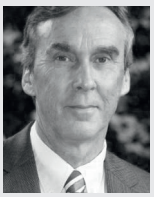
Why Use Ranorex

Watch Now at www.ranorex.com/why

Award-winning test automation tools, which allow testing of many different application types, including: Web browsers (IE, FF, Chrome and Safari), WPF, Flash/Flex, Silverlight, Qt, SAP, .NET, 3rd Party Controls and Java.

Editorial Board

Graham Bath



Graham's experience in testing spans more than 30 years and has covered a wide range of domains and technologies. As a test manager he has been responsible for the testing of mission-critical systems in spaceflight, telecommunications and police incident-control. Graham has designed tests to the highest levels of rigour within real-time aerospace systems. As a principal consultant for T-Systems, the systems division of Germany's leading telecommunications company Deutsche Telekom, he has mastered the Quality Improvement Programs of major companies, primarily in the financial and government sectors. In his current position Graham is responsible for the test consulting group. He has built up a training program to include all levels of the ISTQB certification scheme, including the new Expert Level, for the company's large staff of testing professionals. Graham is the chairman of the ISTQB Expert Level Working Party and co-author of both Advanced Level and Expert Level syllabi.

Gualtiero Bazzana

Gualtiero has been working in the IT and testing domain for the last 20 years; he is author of more than 50 publications on the topics of sw quality and testing; he currently holds the following positions: Managing Director of Alten Italia, Chairman of ITA-STQB, the Italian Board of ISTQB® of ISTQB® Marketing Working Group, Chairman of STF – Software Testing Forum, the most important event on testing held annually in Italy.

Yaron Tsubery

Yaron has been working in software since 1990, and has more than 20 years in testing field as a Test Engineer, Testing TL, and Testing Manager. His original profession was Systems Analyst.
Yaron Tsubery is the current CEO and founder of Smartest Technologies Ltd and has been a Director of QA & Testing Manager at Comverse since 2000 until 2010. Yaron was in charge of a large and distributed group that includes Testing Team Leaders and Test Engineers. The group deals with Functional, Non-functional, Load & Performance tests, some of which are done off-shore. The group has much experience in managing and controlling acceptance tests. Yaron is also known as a manager who has built testing groups in his past and also acted as a consultant to testing managers while they were building their testing groups.
Yaron worked in Product Management, Project Management and Development for at least 3 years before becoming a Director of QA & Testing.
Yaron has wide experience in banking business processes. For the last 10 years he has implemented best practices in a field where he is in charge of producing complex systems for telecommunication companies such as Vodafone, T-Mobile, NTT DoCoMo, Verizon Wireless etc.; Main systems in Mobile technology: i-mode, Mobile GateWay and Billing Proxy; in System Integration projects: Content Data Management solutions including OSS & BSS sub-systems, Video Solutions, IPTV and IMS.
Yaron is the current President of ISTQB® (International Software Testing Qualifications Board – *www.istqb.org*) and is also 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, as well as he wrote articles that were published in professional magazines.

Mieke Gevers

In the IT industry for more than twenty years, Mieke Gevers has developed a special interest in the techniques and processes relating to performance management and automated testing. Mieke is a regular speaker at conferences and organizations worldwide. She served on the Program Committee for the EuroSTAR conference in 2007 and 2009 and was Program Chair of the Belgium Testing Days 2011 and 2012. A co-founder of the Belgian Testers Organization, Mieke has been a board member of KVIV and the ISTQB-affiliate Belgian Software Testing Qualifications Board.

Paul Gerrard

Paul is a consultant, teacher, author, webmaster, programmer, tester, conference speaker, rowing coach and most recently a publisher. He has conducted consulting assignments in all aspects of software testing and quality assurance, specialising in test assurance. He has presented keynote talks and tutorials at testing conferences across Europe, the USA, Australia, South Africa and occasionally won awards for them.
Educated at the universities of Oxford and Imperial College London, he is a Principal of Gerrard Consulting Limited and is the host of the UK Test Management Forum.

Mike Smith

Mike is Managing Director of Learntesting Group (*www.learntesting.com*) and has a broad background in systems development and testing stretching back 30 years.
After starting his career in pharmaceutical research where he continued his education in Chemistry, he moved into IT where he worked in the Telecoms, Pharmaceuticals and Banking sectors.
Mike became an independent consultant in 1984 and initially worked in the development of critical financial applications. He was the original author of the T-Plan Test Process Management tool which was first implemented by The Bank of England in 1989 (*www.t-plan.co.uk*) He also founded ImagoQA Ltd which at the time was the largest independent test consultancy in the UK with global operations in the USA and Australia.
Over the past 20 years, Mike has produced a series of papers about Test Process and Test Management and presented numerous seminars in these subjects together with IT Governance and Information Traceability. Mike was on the presenter list at the very first STAR conference in Las Vegas, 1992 and has also presented at EuroSTAR and StarWEST in more recent years.
Mike is using experience of various entrepreneurial start-ups, board-level appointments as well as his testing and QA background to develop ways in which the profile of testing can be raised in a manner that can be better understood by key business and IT stakeholders.
Mike is Vice-Chairman of the UK Testing Board (which represents the UK on the International Software Testing Qualifications Board – ISTQB) and the Chairman of the ISTQB Advanced Level Syllabus Working Group.
Mike's interests include cookery, wine, cycling, reading, quizzes, sport in general especially cricket and football. Mike was elected a member of the General Committee at Essex County Cricket Club in 2007 following over 10 years of involvement in sponsorship, players' benefits and sub-committees.
msmith@learntesting.com

Erik van Veenendaal

Erik is a leading international consultant and trainer, and a widely recognized expert in the area of software testing and quality management with over 20 years of practical testing experiences. He is the founder of Improve Quality Services BV (www.improveqs.nl) and one of the core developers of the TMap testing methodology and the TMMi test improvement model. Erik is a regular speaker both at national and international testing conferences and holds the EuroSTAR record winning the best tutorial award three times! In 2007 he received the European Testing Excellence Award for his outstanding contribution to the testing profession over the years. He has been working as a test manager and consultant in various domains, including finance, government and embedded software. He has written numerous papers and a number of books, including "Risk-based Testing: The PRISMA Approach", "ISTQB Foundations of Software Testing" and "The Little TMMi". Erik is also a former part-time senior lecturer at the Eindhoven University of Technology, vice-president of the International Software Testing Qualifications Board (2005–2009) and currently vice chair of the TMMi Foundation.
www.erikvanveenendaal.nl

Hans Schaefer

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

Andreas Ebbert-Karroum

Andreas leads the Agile Software Factory (ASF) at codecentric. For more than four years, he is a certified Scrum Master. Since then he could contribute his competencies in small and large (> 300 persons), internal and external, or local and global projects as developer, scrum master or product owner. Meanwhile, he's also a Professional Developer Scrum Trainer, conducting a hands-on training for Scrum team members, which codecentric has developed together with scrum.org and in which he shares with joy the knowledge gained in the ASF. More than 15 years ago, his interest in JavaSE/EE awakened, and it never vanished since then. His focus at codecentric is the continuous improvement of the development process in the Agile Software Factory, where technical, organizational and social possibilities are the challenging, external determining factors.

Agile
TESTING DAYS

October 28–31, 2013
Potsdam, Germany
www.agiletestingdays.com

Call for Papers

We are happy to announce the fifth edition of the Agile Testing Days. Our working title for 2013 is:

"Setting, Increase, Early Veraison and Ripening of Agile Testing"

Just as good wine – Agile testing is maturing and getting better with every added year.

With respect to our conference theme, we would like to receive proposals for track talks of 45 minutes and workshops of 120 minutes duration. Please find our topics of interest below and use the online form on our website to submit your speaking proposal.

The Call for Papers will close after **February 28, 2013**.

Topics of interest:

Test-Driven Development (TDD)/Behavior-Driven Development (BDD) | Acceptance Test-Driven Development (ATDD) | Automated Unit-level Tests | Automated Integration-level Tests | Automated System-level Regression Tests | Automated ATDD | Exploratory Testing on Agile projects | Tools "for Agile Testing": for TDD/BDD; for ATDD; for Continuous Integration (CI), Build and Integration Management, Version Control Systems (VCS); etc. | Refactoring | Emerging design | Managing technical debt in test suites | Creating and maintaing non-brittle test suites | Agile testing and compliance requirements | Agile testing and non-functional requirements | Requirements in Agile projects | Continuous Testing (based on CI): e.g. Automated Regression Testing, Continuous Performance Testing, Continuous Load Testing, Continuous Scalability Testing etc. | Testing and the fast pace of agile development | Measuring Agility | Transition from "traditional" development methodologies to Agile teams | Managing testers on Agile teams | The roles of an Agile tester | Testing and QA | The role of a QA specialist on an Agile team | Collaborative extension: testing challenges for distributed team | Measurable improvements as a result of Agile practices | The problems and solutions of changing to an Agile culture as seen from a tester's perspective | Human perspective and psychologic aspects of Agile development and testing | Collaboration and building teams | Coaching and mentoring | Management and leadership in software organisations | Enterprise Agile | Security Testing | Systems-thinking and patterns | User experience and interaction design | Working with customers | Agile for Embedded Systems

By Alexander Podelko

Performance Testing in the Cloud: Look Beyond the Word

The cloud introduced new opportunities and challenges to performance testing, but specific pros and cons vary significantly depending on your environment and goals. The term cloud is overused and covers a lot of different options. If we want to understand how the cloud may impact performance testing, we should consider all these options separately as they result in completely different performance testing contexts.

In performance testing we have two main components: the system under test and load generators. There may be other components for monitoring, results analysis, etc., but they are not so important in the context of this discussion.

When we talk about load generators, we have three main options:

- Locally, the traditional option (for example, in a test lab).
- As a service. This option has existed for a long time (for example, load testing services provided by Gomez, Keynote, and other companies). While we can refer to it as a SaaS (Software as a Service) cloud now, the only real change is that we now have more such companies (and, respectively, more choices) because it is easier to start such a service using cloud to provide the infrastructure.
- In IaaS (Infrastructure as a Service) clouds. This is a new option which makes it easy to access a large number of remote load generators. It was always possible to have a load generator on a remote machine, but now it is much easier. Some tools provide help with cloud deployments, which may be very handy when you need a large number of load generators for a large-scale test.

When we talk about the system under test, in addition to having the system locally (which may be anything from a development machine to the production system) we may now deploy it in a cloud. This helps to overcome one of the main reasons of not testing full-scale setups – lack of hardware resources. Now you can access as much resource as you like when you are ready for it. However, it may not be exactly the same kind of hardware and software that you use in your production system, so in getting closer to the scale of the system you may be farther away in terms of the details of the environment.

The best combination of options for you depends on the goals of performance testing. Performance testing in the cloud (or from the cloud) makes sense for certain types of performance testing. For example, it should work very well if we want to test how many users the system supports, whether it would it crash under a load of X users, or the number of servers we need to support Y users, but only if we are not too concerned with exact numbers or variability of results (or even want to see some real-life variability).

Even in this case, the assumptions are that we aren't introducing any bottleneck using the cloud (for example, saturating network bandwidth between the load generators and the system under test) and leave it to the cloud provider to worry about whether our tests will impact other cloud tenants.

However, it doesn't work well for performance optimization, when we make a change in the system and want to see how it impacts performance. Testing in a cloud with other tenants intrinsically has some results variability, to the extent that we don't control other activities in the cloud and, in most cases, don't even know the exact hardware configuration. For example, if the system scales out by automatically creating an additional application instance, the new instance may be outside of the network segment where other servers are. The effects may be even more sophisticated in case of Platform as a Service (PaaS) or SaaS clouds. So, when we talk about performance optimization, we may still need an isolated environment.

One interesting case is when the system is created to be used in a cloud, which probably will become more and more common over time. The first thought would be that it simplifies the choice – you just test it in the cloud where it is supposed to be deployed. Still, it will not work too well if you need to do performance optimization or troubleshooting and want tests to be completely reproducible. In this case, you may need something like an isolated private cloud with hardware and software infrastructure similar to the target cloud, and monitoring access to the underlying hardware to see how the system maps to hardware resources and if it works as expected. Real-world network emulators may be used to make sure that performance testing is representative of how the system would be used in production – otherwise we would not be taking into account such factors as network latency, bandwidth, jitter, etc. So, if we need optimization for cloud software, we may still need a lab – but the lab should be more sophisticated to emulate the cloud environment and real-world network conditions. An ultimate example of such a lab is the lab Microsoft created for testing IE, described at *blogs.msdn.com/b/b8/archive/2012/02/16/internet-explorer-performance-lab-reliably-measuring-browser-performance.aspx*.

Thus, we have different options for the system and load generators deployments, and which option (or combination of options) would be the best depends on the goals of performance testing. For example, some typical performance testing scenarios might be:

- System validation for high load. Outside load (service or cloud) against the production system may be the best option here. We have a wider scope of testing, but lower repeatability.
- Performance optimization/troubleshooting. An isolated environment may be the best option here. We have a limited scope, but high repeatability.
- Testing in cloud. This may be the best option for periodic tests to lower costs. We have a limited scope and low repeatability.

So, factoring the cloud into performance testing, we have at least two major alternatives (with a variety of subtler options): coarse performance testing in or from the cloud with inherent variability (and probably some savings on hardware and configuration costs) or granular performance testing and optimization in a isolated environment (thus avoiding variability with, probably, higher hardware and configuration costs). For comprehensive performance testing you may even need both lab testing (with reproducible results for performance optimization) and realistic outside testing from around the globe (to check real-life issues that you cannot simulate in the lab). Doing both would be expensive and makes sense only when performance really matters – but if you are not there yet, you may get there eventually. ■

> about the author

For the last fifteen years, **Alex Podelko** has worked as a performance engineer and architect for several companies. Currently he is Consulting Member of Technical Staff at Oracle, responsible for performance testing and optimization of Hyperion products. He blogs at *www.alexanderpodelko.com/blog* and can be found on Twitter as *@apodelko*.

Enjoy Test Case Design in the Cloud

CaseMaker SaaS systematically supports test case design by covering the techniques taught in the ISTQB® Certified Tester program and standardized within the British Standard BS 7925. The implemented techniques are: Equivalence Partitioning, Boundary Check, Error Guessing, Decision Tables and Pairwise Testing. Risk-Based Testing is supported as well.

CaseMaker SaaS fits between
Requirement Management and
Test Management/Test Automation.

Subscribe and try for free!
Decide afterwards.

One license starts at

75 €/month (+ VAT)
saas.casemaker.eu

PayPal

CaseMaker® SAAS

By Kari Kakkonen & Heikki Uusitalo

Personal Testing Process (PTP) – Individual Level Development

We were inspired a few years ago at Knowit (then Endero) by Watts Humphrey's Personal Software Process (PSP). We were also influenced by the continuous learning idea of exploratory testing and software process improvement frameworks such as SPICE (ISO15504), CMMI, and, naturally, the Bloom knowledge level taxonomy. Testing consultant Heikki Uusitalo received an assignment to create a new methodology with the support of Kari Kakkonen and to pilot the methodology in customer projects. As a result, the Personal Testing Process (PTP) was created. We observed that PTP works really well at individual level development, so we trained the rest of the consultants in the company. PTP has been used widely since then. It is an excellent way to think and a way for testers to make their actions more efficient.

Bloom's taxonomy defines the levels of knowledge as

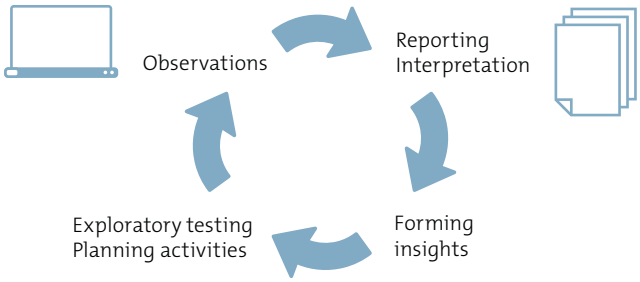
- **Remember.** Show that you know.
- **Understand.** Show that you understand.
- **Apply.** Show that you can utilize what you have learned.
- **Analyze.** Show that you can find the relevant information.
- **Synthesize.** Show that you can create new elements by combining existing ones.
- **Evaluate.** Show that you can assess ideas, information, processes, and solutions.

1. Personal Testing Process – PTP

PTP is a staged model developed for testing information systems. The model measures the capabilities and maturity of the tester. Each tester can use the model to define his/her individual level. By following the level definitions and fulfilling the requirements of the next level, a tester can raise to that next level.

The basic idea is that the tester acts and works in regular ways. Each tester creates a process out of their own activities, then they develop further and improves mainly on the basis of the accumulated experiences.

To accomplish this, the tester must be able to observe their work as a process in which tasks and activities are defined. Then each activity can be measured to make the activities comparable, which is the prerequisite for developing the process.



Adapted from source: *Psychology of Usability*, Sinkkonen et al.

The model includes five levels. It starts from the basic execution of tests, where a personal testing process is created. This process is then further improved on the basis of the measurement results.

2. Organization and Tester Process

An organization has a testing process which is the way the organization performs testing. The process does not usually define how individual testers should go about executing their testing activities, instead the tester has freedom to perform the agreed activities within the boundaries defined by the testing process. This freedom is the area where personal testing process belongs.

When a tester find during their work that there are activities not defined by the organization's testing process, it is a fruitful time to consider these activities as part of the personal process. The activities and their execution rely on the tester's own instructions.

When the tester learns how to recognize the repetition of similar work tasks in their work and to choose and create a consistent way to perform these tasks, they also automatically create the potential to improve this process.

Then the tester starts working consistently and the results of the tester's work are also improved overall. When an individual tester's results are improved, this also affects the results of the whole organization.

3. Five Levels

Tester is on level:

- **One:** Can test. (Understanding)
- **Two:** Manages test execution. (Apply)
- **Three:** Can establish personal process. (Analysis)
- **Four:** Measures and analyzes the process. (Synthesis)
- **Five:** Sets goals and develops process. (Evaluate)

The Personal Testing Process model is defined as five levels. Each level has its own attributes, based on the personal level defined for the tester. When a tester fulfills all the requirements of the level, they are at that level.

3.1 Level 1

At level 1, the tester knows the basics of testing and can work independently. The tester can create a test case, execute a test, make a decision about accepting or rejecting a test, they will know how to write an incident report, and draw a conclusion about test executions. The tester also knows the software development process.

3.2 Level 2

At level 2, the tester can perform testing effectively, utilize testing tools and justify their benefits, measure test executions, understand information system definition and modeling techniques, and they know how to estimate and review testing work.

3.3 Level 3

At level 3, the tester recognizes regular, repeated tasks and creates a personal process out of them. The tester make their work consistent and can observe their activities as a process. The tester can measure the process and improve it based on the measurement results.

3.4 Level 4

Level 4 has two parts:

- The tester may require good ways of working and documentation from the developers of the system.
- At the same time, the tester improves their ways of working. They aim to be involved in system planning (definitions, modeling, and design) and to provide test requirements as part of the planning work.

Principal requirements at this level are the analysis of measurement results and the improvement based on the analysis. Testers must be able to review, utilize different methodologies in process development, and optimize and minimize processes.

3.5 Level 5

At level 5, the tester analyzes the process, sets improvement targets, finds ways to carry out the improvements, and is able to change the process.

4. Processes

4.1 General Testing Process

The general (organizational) testing process refers to a model that fits almost all the different testing tasks. The model can be based on different life-cycle models, such as the Agile process or incremental process, and it is different from organization to organization. This kind of model is used in the context of this document to help create a personal model.

Some requirements of the general process are: requirements definition, test planning, test case design, test execution, reporting, test results analysis, retesting of fixed incidents, regression testing, and closure of testing.

Below is an example from a plan-driven testing process which aims to help testers recognize the activities of a process and name their own tasks in the process. Testers choose one activity at a time. For each activity, they consider whether the activity has tasks in which they need to make independent choices or decisions. They then repeat this analysis until the whole process (its activities) has been performed. At the same time, the suitable candidates for the personal process are identified or created.

4.2 Personal Testing Process

With the help of the general process, testers find their tasks and they can see their activities as a sequence of tasks. Based on this understanding, testers create their personal process by defining tasks in such a way that they form a continuous stream of tasks – a process.

The process is created when it is possible to write a procedure about the testing which can be used as the basis for performing testing.

Test planning, design, and execution are the key activities where the personal process can be seen most clearly.

When test case design and implementation are performed using the same sequence of activities, the tester effectively has a guide through the process. The test case designer knows what kind of tests they need to create and can estimate the workload of this. The test execution is clear when the whole process is known and it is possible to anticipate which task will be next.

To continue with the plan-driven example, the focus in test case design becomes the singularity of test case procedure or individual instructions. The test cases are written in enough detail so there are no interpretation difficulties. This can be achieved, for example, by modularizing the test cases into smaller parts until the singularity is achieved.

5. Process Creation and Development

When the process is defined, it is important to remember that the process received inputs, out of which it produces outputs. To be able to create outputs, the process needs resources and other supporting activities. Also, different guidance factors affect the process. These four streams (input, output, support, guidance) must be taken into account.

- In the beginning, testers define their personal needs and priorities. They answer questions about why they are doing this. Is the objective to be more effective, efficient, fault-free, or some other quality characteristic?
- Next, the tester defines process targets, objectives, and quality requirements, i.e. Where do we focus the activities? What are we trying to reach with the activities? And with which criteria?
- When the first two steps are clear, the tester defines the current process, i.e. how the tester currently works.
- When the current process has been defined, a target process can be defined. This is achieved by changing the current process in such a way that the objectives and quality requirements are met.
- After this, the tester creates a development strategy which leads us to the target process. The tester defines what to do and in which order.
- The tester creates the actual process.
- The tester validates the process by using it and observing that it works.
- The functioning process will meet continuous change pressure for internal and external reasons. These will lead to continuous process development.

5.1 An Example

A simple example is to consider different ways of testing two fields in a user interface. Each field can be subjected to positive and negative testing. The fields can be tested in different orders. The tester decides that their personal way of testing this is to first perform both positive and negative tests on one field and then repeat the routine on the second field. The alternative would be to first execute positives tests to both fields and only then the negative tests to both fields. When we think about this decision from the perspective of the general testing process, we see that the decision affects almost all process activities. First, the tester designs the positive tests and only then the negative tests. The tester first implements the positive tests and then the negative ones, with the same thing happening in the execution schedule (positive tests in the morning and negative tests in the afternoon), etc. This small decision affects the whole chain of activities. The important part is that the tester knew before starting the activities what and how they will do things. This has a great impact on productivity.

5.2 Benefits

The Personal Testing Process is a (new) way of thinking about things. It is a tool or methodology to help improve the professionalism of the tester. Its objective is to improve quality and productivity. When personal productivity increases, the organizational productivity will also improve. Personal process development can also be used to help improve the competencies of the tester and even to help salary and career development. ■

> about the authors

Kari Kakkonen has a Technology MSc in Industrial Management from Aalto University and has the ISTQB Full Advanced certificate. He has been working in software testing and quality assurance since 1996, working as a test manager, project manager and service manager, as well as consulting, creating testing processes and organizations, and delivering training. Kari has worked for various industries and companies, including in banking and finance, pensions insurance, telecoms, embedded software, public sector, information systems, and commerce.
Kari Kakkonen is currently working in Finland at Knowit, a Nordic ICT services company specializing in testing consultancy and software development. Kari delivers training in Agile testing, exploratory testing, test process development and the ISTQB Advanced courses, among other things. Kari usually gets almost the maximum score in training feedback and is one of the most highly appreciated trainers in Finland. Kari has delivered well over 1000 person training days, a total of almost 150 actual training days. Kari is chair of the Finnish Software Testing Board (FiSTB) and a member of the ISTQB expert level working group. He is also on the board of the Finnish Association of Software Testers and is included in *Tietoviikko* magazine's 100 most influential people in IT 2010 listing. Kari has written a handful of articles for various software testing magazines on subjects such as software testing, processes, distributed projects, plan-driven and Agile differences.

Heikki Uusitalo has a BSc in Information Technology from Helsinki University and holds the ISTQB Foundation certificate. He is an experienced testing consultant who has worked in all kind of software development since 1974, starting his career as a programmer in banking systems.
Heikki Uusitalo works in Finland at Knowit as a testing consultant. He is also the chair of the Testing Models and Standards group of FiSMA (the Finnish Software Measurement Association).

ISTQB® CERTIFIED TESTER ADVANCED LEVEL

TM TA TTA

ALL GOOD THINGS COME IN THREES

FOR THE PRICE OF 2

For more than ten years, the International Software Testing Qualifications Board (ISTQB®) has been providing the most renowned certification scheme for software testers.

To celebrate this anniversary, iSQI is offering all three Advanced Level certification examinations for the price of two. Become "FULL ADVANCED" now!*

Register your certification examination today at www.isqi.org/en/registration.html

registration till January 2013

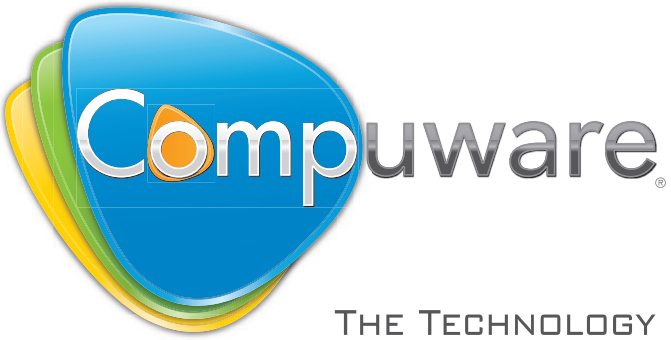
BOOK YOUR ISTQB® FULL ADVANCED LEVEL CERTIFICATION NOW AND GET ONE ADVANCED LEVEL EXAM FOR FREE!*

TM TA TTA

*OFFER VALID FROM 19 SEPTEMBER 2012 UNTIL 19 JANUARY 2013, CERTIFICATION EXAMINATION MUST BE TAKEN BEFORE 31 DECEMBER 2012 / THIS OFFER ONLY APPLIES TO YOUR FIRST ATTEMPT! / THIS OFFER CANNOT BE USED IN COMBINATION WITH ANY OTHER DISCOUNTS. ALL PREVIOUS CERTIFIED TESTER CERTIFICATIONS MUST HAVE BEEN ACQUIRED THROUGH ISQI / PLEASE CONTACT US WITH WRITTEN PROOF OF YOUR CERTIFIED TESTER FOUNDATION LEVEL STATUS AND / OR PROVE OF ANY CERTIFIED TESTER ADVANCED LEVEL (TM, TA,TTA) PREVIOUSLY ACQUIRED.

CERTiFYING PEOPLE
www.isqi.org

iSQI® International Software Quality Institute



By Michael Kopp, Technology Strategist, Compuware

The Next Level of Application Performance Management (APM) in the Cloud

New APM Dimension Emphasizes Cost-Per-Transaction; Helps Testers Elevate Strategic Role in Maximizing Cloud Investments

Businesses expect many benefits when they move to the cloud, including greater business agility, significant cost-savings and of course, increased profitability. Continuous application performance management (APM) supporting the entire application lifecycle can be the key to achieving these benefits, in more ways than one.

Recently, there has been a lot of discussion around the importance of APM in the public cloud, particularly as it pertains to ensuring fast, reliable end-user application experiences. The cloud is opaque, meaning that businesses using the cloud often have little insight into the inner workings and capacity management decisions of their chosen cloud service provider. There's only one way for businesses to know for sure that end users are having a fast, reliable experience with a cloud-based application. Testers need to measure this performance from the true end-user perspective on the "other side" of the cloud.

Because they are on the "front lines," testers are often the first to determine when a performance problem has occurred in a cloud-based application. Deep-dive diagnostics can then help identify the source of a performance problem, whether it's the cloud, the business' own data center or another element in the application delivery chain. If the source of the problem is in fact the cloud, businesses can then leverage this valuable information to alert the cloud service provider and uphold application performance-focused service level agreements (SLAs).

Many businesses and their testers have made significant progress in using APM to ensure a high level of cloud application performance. They may have reached a point that their applications are fast and reliable enough, but this is no time to stop doing APM in the cloud. There's tremendous benefit to be gained by having a better understanding of the inner workings of cloud-based applications. The next level of APM in the cloud is all about optimizing the cost structure of a cloud-based application, which, just like application performance, has a direct impact on cloud return-on-investment. Once again, testers have a strategic opportunity to help maximize cloud investments and boost the bottom line.

Put another way, where the cloud is concerned, APM isn't just about making applications faster. It's also about making applications – particularly the highest-volume applications – more efficient from a cost perspective. This needs to be taken into consideration at every step of the application development process, because even the supposedly best application is never complete. Ongoing enhancements, fixes and optimizations can dramatically impact both the performance and cost-efficiency of a cloud-based application. Therefore, both aspects must be considered throughout regression testing and the entire application lifecycle.

Consider a search function – in addition to being fast, testers need to optimize it so that it delivers better results and is not executed five or more times by every user. This can be considered functional optimization, but also lowers the operational cost, because in the cloud every transaction has a dollar value attached to it. Making less database access calls per search transaction, while not making the search faster at all, can save money. This is because most cloud providers charge on a per operation – e.g. per SQL – basis. Thus optimizing the number of SQLs might be more cost effective than saving CPU! In this approach, businesses completely sidestep the question of resource optimization and go straight to where it matters – cost optimization – and testers can really lead this charge. The fact is, businesses don't really need to care about resource usage in a public cloud at all. They need to care about the true end-result – application SLAs and cost effectiveness – and increasingly, testers are the purveyors of this knowledge.

Another example is the purchase function. Are there any features that consume a lot of resources in the cloud – e.g. product tours or images – that end-users spend a lot of time on? By knowing how end users are spending their time, testers gain key insights into what may be driving up costs in the cloud. Furthermore, by understanding the cost structure of a transaction and how much revenue it generates, testers are better able to set priorities so transactions leading up to conversion can be optimized functionally, cost-wise and performance-wise.

For many businesses, one of the primary perceived advantages of moving to the public cloud is elastic scaling. Elastic scaling avoids the old way of capacity planning and big expenses upfront; instead businesses can increase the size of their environments as their load increases. But elastic scaling also has a disadvantage. Over-consuming planned capacity can happen easily since there is no hard limit, which often leads to exceeding cost-estimations.

It's therefore critical for testers to directly understand how end users interact with an application – where APM in the form of end-user experience monitoring comes in – and how the application handles the load. In this way, testers can offer valuable information that helps guide more informed, judicious cloud capacity decisions. Not having this information is operational blindness and comparable to simply passing on the company credit card.

In summary, end-user experience management enables testers to understand end users' behavior and how performance affects conversion rates and the business. But in a public cloud this is only half of the APM story. Only if businesses can accomplish both – satisfy end-user expectations for fast, reliable applications while keeping costs down – can they be successful in the cloud and improve their business performance as a result of using the cloud. By focusing on cost-per-transaction, testers can increase their strategic contributions to business profitability. ■

> about the author

Michael Kopp is a technology strategist in the Compuware APM center of excellence and has more than 10 years of experience as an architect and developer in the Java/JEE space. Additionally, Kopp specializes in architecture and performance of large-scale production deployments with a special focus on virtualization and cloud.

By Sujith Shajee

Simplifying Testing Cloud and Cloud Testing

Introduction

"Testing on cloud", is a relatively confusing phrase, not from the conceptual point of view but due to the expression's wording. The question that this phrase usually leaves in the mind is: "Does this term signify *testing with the help of something that is in the cloud*, or simply *testing something that is in the cloud*?"

As every other blog, paper or discussion around this concept would unanimously agree, cloud is not a new concept but has been around in business for many years now. Cloud has received all the acclaim for bringing the concept of on-demand into the IT industry.

Breaking down the testing paradigm into "testing cloud" and "cloud testing" will make a really good start towards understanding and simplifying the concept

Testing Cloud

Let us break down this concept using a very simple example. A web service intended to perform amortization calculations for mortgage loans based on the loan amount, APR and tenure is deployed on cloud infrastructure and is opened up to consumers. This web service is used by many financial institutions offering Mortgage loans as part of their range of services.

Taking into consideration factors concerning the user base using the web service, diverse test strategies come into play. Assuming the web service was functionally stable when the deployment was done, the need to test the web service for performance on the cloud is still critical.

Let us look at the various testing strategies which come into play when validating the above web service after it has been deployed on a cloud vendor's cloud infrastructure to determine stability

- **Load testing** – Increase demand on the web service and determine the response time on the infrastructure provided by vendor.
- **Stress testing** – Increase load gradually and determine the break point of the web service on the infrastructure provided by vendor.
- **Availability testing** – Run the web service for a prolonged time to determine any failure events and recovery time for the vendor.
- **Security testing** – Perform this to determine any authorization or authentication failures with the cloud vendor.
- **Interoperability testing** – Determine any issues when environment migrations are done.
- **Latency testing** – Determine any anomalies as part of sending the request over to the cloud and receiving the response back over the cloud.
- **Endurance testing** – Determine any issues with constant load on service over a period of time.
- **Spike testing** – Induce load spikes and determine any downtimes or performance issues.
- **Scalability testing** – Determine scale-up and scale-down capabilities of cloud vendor.

The point here is that, when validating cloud, a quality assurance engineer is trying to determine the stability of the infrastructure provided by the cloud vendor with the application that is to be hosted there.

Conclusion

- Testing cloud is intended to test applications deployed on the cloud.
- Apart from functional validation, the application is validated for performance and security when deployed on the cloud.
- The intention is to performance test the cloud infrastructure during the validation process on the cloud.

Cloud Testing

Now it is time to figure out what the term "cloud testing" actually means.

But first, let us talk about cloud as a service, i.e. delivering what the customer wants as a service over the network. This service could be infrastructure-based, software-based, platform-based or even storage-based. So, at the end of the day, the service helps the customer to avoid the overheads and cost that come as part of maintaining the service.

For instance, an organization may not be able to afford the expense of procuring software such as Microsoft Office, Google Apps and so on for its day-to-day operations. This software comes with own overheads of maintenance and upgrades. Using Software as a Service (SaaS), the organization can avoid the hassle involved in maintaining infrastructure and platforms to run the applications. All upgrades and maintenance are managed by the cloud vendor which the organization chooses to use.

Cloud testing uses infrastructure available on the cloud to perform validation activities. One of the main reasons why organizations adopt cloud testing is to avoid the high cost incurred in maintaining testing tools. Limited funds for tool license procurement and strict deadlines are other factors that lead organizations to opt for cloud testing. One of the other attractive aspects of cloud testing is its on-demand nature. To put it in layman's terms, you pay if you use it.

Cloud testing vendors provide customers with a variety of services including testing software for automation and performance testing, infrastructure that will simulate production environments, and platforms that help quality engineers develop automation scripts for applications testing.

Let us look into a few characteristic features of implementing cloud testing.

- For the customer, the first and main factor to be reduced is the cost of maintaining infrastructure and software. Since these responsibilities lie with the vendor, there is an assured monetary benefit for the customer.
- Another important feature is the on-demand service. You pay for what you use and that makes things simpler at all levels.
- The customer does not have to worry about any upgrades and maintenance to the software being used. Regular checks and upgrades are the responsibility of the vendor.
- Since the resources are shared across a larger group, resource utilization is assured.
- Infrastructure is centralized at one location, thus lowering cost.
- Provisioning of resources provides control over the resource sharing and enables scalability without experiencing load peaks.
- Lower environment setup time.

Having said this, there is a flip side to this coin. Cloud testing comes with its own issues such as security, initial set-up cost, and so on. Another point on the debit side when evaluating cloud is whether or not the performance on the cloud infrastructure can be accurately validated and that it may vary due to the performance of the cloud vendor's infrastructure. However, these are some issues which can be contained by the customer through careful study and handling.

Conclusion

- Cloud testing is the result of cloud as a service.
- Testing infrastructure or software hosted on cloud is leveraged for testing.
- The intention is to leverage what is out there on the network for cost effective testing.

Afterthought

Testing cloud and cloud testing are entirely separate entities with their own responsibilities in the IT industry. While the former is orientated towards getting the application established on cloud infrastructure, the latter is about leveraging the infrastructure on the cloud and validating the application. ■

> about the author

Sujith Shajee is currently working as a Test Analyst at Infosys Limited (NASDAQ: Infy *www.infosys.com*) and is part of the Independent Validation and Testing Services unit. He has worked on various projects from the strategy and delivery point of view, has handled automation, performance and service validation projects, and has expertise in a number of validation tools. Sujith can be reached at *SujithK_Shajee@infosys.com*.

The ASQF Career Center

We've got the right job for you.

- **Consultants (m/f) for model-based Softwaredevelopment** — *MID GmbH, Köln, Nürnberg, Stuttgart*
- **Development Engineer (m/f) in full time** — *tecmata GmbH, Wiesbaden*
- **Experienced Experts and comitted Young Professionals** — *SQS AG, Köln*
- **Junior Test Manager (m/f) Finance und Banking IT** — *Talentschmiede.com, Rhein-Main Gebiet*
- **Software Tester / IT Quality Management Assistant (m/f)** — *affilinet GmbH, Hannover*
- **Software Test Engineer (m/f)** — *BDC EDV-Consulting GmbH, Wien*
- **Software Developer Web-Application** — *CONTACT Software GmbH, Bremen*
- **IT-Specialist (m/f)** — *sepp.med GmbH, Röttenbach*
- **Software Developer (m/f)** — *GFB EDV Consulting und Services GmbH, Oberursel*
- **Software-Tester (m/f) Quality Management** — *GFB Softwareentwicklungsgesellschaft mbH - Q-up, Oberursel*
- **Consultant Test Data Management (m/f)** — *GFB Softwareentwicklungsgesellschaft mbH - Q-up, Oberursel*
- **Software Engineer (m/f) .NET** — *GFB Softwareentwicklungsgesellschaft mbH - Q-up, Oberursel*
- **Tester/Testdesigner (m/f)** — *T-Systems Multimedia Solutions GmbH, Dresden oder Rostock*
- **Software Consultant (m/f)** — *F+S Fleckner und Simon Informationstechnik GmbH, Limburg*
- **Software Developer (m/f)** — *F+S Fleckner und Simon Informationstechnik GmbH, Limburg*
- **Web Application Engineer (m/f)** — *ReliaTec GmbH, Garching*
- **Software Tester - Test Automation (m/f)** — *DEMIRTAG Consulting GmbH, München*
- **Software-Tester – ISTQB Certified Tester (m/f)** — *DEMIRTAG Consulting GmbH, München*
- **Trainer (m/f)** — *DEMIRTAG Consulting GmbH, Augsburg*
- **Consultant/System Engineer Infrastructure Services (m/f)** — *SOGETI Deutschland GmbH, Hamburg*
- **Software Tester - Test Automation (m/f)** *SOGETI Deutschland GmbH, Hamburg, Düsseldorf, Frankfurt, Stuttgart und München*
- **Software Tester - Testmanager (m/f)** *SOGETI Deutschland GmbH, Hamburg, Düsseldorf, Frankfurt, Stuttgart und München*
- **Software Tester (m/f)** *SOGETI Deutschland GmbH, Hamburg, Düsseldorf, Frankfurt, Stuttgart und München*
- **Software Tester - Last & Performance (m/f)** *SOGETI Deutschland GmbH Hamburg, Düsseldorf, Frankfurt, Stuttgart und München*
- **Software Tester - Mobile Applicationen (m/f)** *SOGETI Deutschland GmbH, Hamburg, Düsseldorf, Frankfurt, Stuttgart und München*
- **Software / System Tester – Automotive (m/f)** *SOGETI Deutschland GmbH, Hamburg, Düsseldorf, Frankfurt, Stuttgart und München*
- **Heros of C# - Development (m/f)** — *CLEAR IT GmbH, Erlangen*
- **Good Nose in Software Debugging (m/f)** — *CLEAR IT GmbH, Erlangen*
- **Barista for a perfect in Form Java Code (m/f)** — *CLEAR IT GmbH, Erlangen*
- **Assistant (m/f) Division functional Systemtest** — *Accenture GmbH, Kronberg im Taunus*
- **Software Quality Engineer (w/m)** — *CAS Software AG, Karlsruhe*
- **Test Engineer (m/f)** — *Stryker Leibinger GmbH & Co. KG, 79111 Freiburg*
- **Software Testanalyst (m/f)** — *Océ Printing Systems GmbH, Poing*
- **Software / System Test Engineer (m/f)** — *Valyue Consulting GmbH, Stuttgart*
- **IT-Consultant (m/f)** — *modulo3 GmbH, Nürnberg/Region Oberfranken*
- **Consultants** — *KUGLER MAAG CIE GmbH, Kornwestheim*
- **Consultant & Trainer (m/f)** — *SOPHIST GmbH, Nürnberg*
- **Microsoft TFS Specialist (m/f)** — *CMCS GmbH, Erlangen/Nürnberg*
- **Consultant SW-Quality Management (m/f)** — *Software Quality Lab GmbH, München, Wien oder Linz*
- **Manager Tool Evaluation Center (m/f)** — *Software Quality Lab GmbH, München, Wien oder Linz*
- **Software Test Engineer (m/f)** — *Software Quality Lab GmbH, München, Wien oder Linz*
- **Testmanager (w/f) with Banking Know How** *ISMC Information System Management & Consulting GmbH, München, Frankfurt oder Karlsruhe*
- **Software Tester (w/f) with Banking Know How** — *ISMC Information System Management & Consulting GmbH, Waldbronn*
- **Testmanager (m/f)** — *R+V Allgemeine Versicherung AG, Wiesbaden*
- **Consultant Digital BOS Radiocommunication (m/f)** — *IABG mbH, Ottobrunn und Berlin*
- **Senior Consultants IT Management & Quality Services** — *Díaz & Hilterscheid Unternehmensberatung GmbH, Berlin*
- **Consultant - Specialist SCM, ALM und Test Automation** — *elego Software Solutions GmbH, Berlin und Dresden*
- **Basic-Software Developer LIN (m/f)** — *Brose Fahrzeugteile GmbH & Co. KG, Coburg*
- **Software Engineer (m/f) AUTOSAR** — *Brose Fahrzeugteile GmbH & Co. KG, Coburg*
- **Software Architect (m/f)** — *infoteam Software AG, Dortmund, Erlangen und Zürich*
- **Software Developer (m/f) - Java** — *infoteam Software AG, Dortmund, Erlangen und Zürich*
- **Software Developer (m/f) - C#, .NET** — *infoteam Software AG, Dortmund, Erlangen und Zürich*

Detailed information on all job offers on **www.asqf.de/stellenangebote.html** *(German only)*

By Massimo Forno

Software Testing: Adapt and Grow with the Cloud

Cloud computing has certainly brought a lot of new possibilities in terms of business and, clearly, as the arrival of a new technology or methodology often requires, software testing needs to adapt and grow in this direction.

Several new testing services are already available and others will follow shortly. This helps us to understand who has benefitted the most from this type of computing. An example of this is the software testing companies which, aside from the value proposition offered by these new services, can now reduce their cost of testing by taking advantage of multi-tenant and off-premises architecture. This helps them to avoid the purchase of expensive hardware and software and, at the same time, allows them to scale faster and less expensively.

Moreover, configuring the test environment has become easier and almost instant, allowing for faster test execution which has the potential to increase the number of test sessions that can be run in the same time period.

As mentioned earlier, companies whose core business is the provision of testing services can exploit this new architecture in ways that result in clear benefits for software vendors. In fact, their goals often coincide since their main concerns are typically related to costs and budget. On the one hand, software houses tend to invest more in business analysis and development activities, asking test teams to be both effective and efficient with limited resources. On the other hand, testing providers want to maximize their revenues by trying to reduce costs. This is especially true in those countries where less effort is invested in testing due to psychological and economic barriers, resulting in compulsory cost savings policies.

This article examines how Cloud-based testing is positively affecting several types of testing.

Functional Testing is probably the least affected type of testing, except in the case of test outsourcing where testing companies are asked to take charge of non-web based applications, which require direct access. This situation is a little more complicated because, even if the issue of direct access can be resolved by using *Remote Desktop Services* to access the software house test environment, or by allowing the testing company to directly deploy the application in their internal test environment, software vendors are not always willing to grant access to their network to third parties and/or share the application code.

In this case the Cloud can help a lot by providing quicker and safer access to the application subject to test.

Non-Functional Testing is more complicated and certainly the most affected, firstly because of the different test types (Performance, Security, Compatibility, Usability, etc.) and, secondly, because it usually requires a huge investment in hardware and software licences.

Performance testing (including **load** and **stress testing**) probably gets the biggest benefit from Cloud computing, considering both the economic aspects and the needs of scalability in order to gradually increase the simulated load.

Using the Cloud means paying only for what is needed and only for the required time. This means that it is no longer necessary to purchase expensive annual licenses for two or three months of testing per year. Server provisioning in the Cloud is almost immediate and decommissioning means that there are no problems associated with reallocation. Furthermore, the test environment can be shared with the development teams, which inevitably helps to improve the communication between testers and developers.

With this configuration, testing companies can maximize their revenues because their cost of testing decreases proportionally, bearing in mind that they are no longer obliged to buy expensive hardware and software, ensure business continuity, or manage frozen hardware and software requirements.

On the other hand, software vendors now have a bigger range of options because many testing providers have already moved their services to the Cloud, often offering test agents and controllers ready-to-run homemade scripts. This still means that software houses can decide which logic and functionalities to test without having to worry about infrastructure costs

and configuration. This allows them to use on-demand testing tools that would otherwise be very expensive.

Security testing executed in the Cloud provides significant benefits. Firstly because many aspects of this type of testing are computing and storage intensive and, secondly, because Cloud computing can be considered as the natural environment where both periodic and occasional activities can be executed. This allows accurate and actionable results to be obtained whilst only paying for what is needed for the required time.

Moreover, as this is managed by IT specialists, Cloud usage is usually safer and more efficient when compared to traditional internally managed test environments.

Compatibility testing is another type of testing that demonstrates how this architecture can reduce costs by avoiding the need to buy licences for multiple operating systems (and different versions of those operating systems).

Cloud-based testing can also help to speed up the creation of multiple operating system and browser combinations, even when they are managed using virtualization techniques.

Conclusion

From the software solution provider's perspective, independent of whether or not a company has its own *Software Quality Control Unit* (and/or test teams), there are clear economic and performance advantages to using Cloud-based testing.

Testing provider companies are taking increasing advantage of Cloud usage, especially for those services that require a big infrastructure in order to perform test activities, particularly for non-functional testing.

To summarize the major benefits:

- Cost savings both for software vendors and testing providers
- Easier and faster test environment setup
- Easier, faster and cheaper resizing of the test environment
- Faster test execution which allows more test sessions to be executed
- Increased quality of the results due to the IT specialists managing the Cloud ■

> about the author

Massimo Forno is Head of Software Quality Control at deltatre S.p.A., Turin, Italy and the CEO of IxmaSoft, Turin, Italy. He has more than 10 years of experience in the software testing field, especially in the area of test analysis and management of complex websites and web applications.
He has advanced knowledge of Agile testing, mobile testing, test automation and test policies/strategies definition. He acquired his experience working on projects in industries such as sport, banking and finance, health, telecommunications, and e-commerce. Feedback regarding the article would be very much appreciated by email: *Massimo.Forno@ixmasoft.it*

BELGIUM TESTING DAYS

FEBRUARY 27TH · MARCH 2ND

Pull down the walls for boosting up the business!

Sheraton Brussels Airport Hotel, Belgium
February 27 – March 2, 2013

Belgium's most popular international testing conference.

With the world's greatest testers, handling topics like Test Automation, Security, Agile, Cloud, Fuzzing, Performance, Mobile, Outsourcing, Metrix, Communications, Agile, Scrum, Advanced workshops and much more.

A total of 2 days challenging conference, 2 days inspiring tutorials and amazing networking events.

Register now at:

www.belgiumtestingdays.com

With 55 speakers from all over the world:
Alexander Podelko, Alon Linetzki, Andreas Faes, Andreas Grabner, Antti Häyrynen, Ard Kramer, Astrid Notø Larsen, Bart Van Ginderdeuren, Bernd Beersma, Björn Vanhove, Bryan Bakker, Chris Van Bael, Debra Friedenberg, Derk-Jan de Grood, Dorothy Graham, Doug Hoffman, Elalami Lafkih, Erik Bits, Erwin Pasmans, Fiona Charles, Geert Colpaert, Geert Peeters, Gerie Owen, Goranka Bjedov, Gumpu Ravi Kumar, Huib Schoots, Jean-Paul Varwijk, Jeanne Hofmans, Jerry E. Durant, Joeri Wijns, Julian Harty, Kris Laes, Kristian Trenkel, Lee Copeland, Leo van der Aalst, Lisa Crispin, Luciano Floridi, Maarten Van Eyken, Manjunath Ramachandra, Marc Rambert, Mark Fewster, Oana Petrascu, Peter Morgan, Peter Varhol, Raluca Popescu, Rik Marselis, Rob Sabourin, Robert Allewijn, Sami Söderblom, Steven Wierckx, Stuart Reid, Susan Windsor, Thomas Sundberg, Vicky De Roeck, Zuzi Sochova

Gold Sponsors

coralius

Compuware
THE TECHNOLOGY PERFORMANCE COMPANY

SOGETI

NEOTYS

Microsoft

Díaz Hilterscheid

Silver Sponsors

iSQI
International Software Quality Institute

ps_testWare
Your devil's advocate

By Mithun Kumar S R

Victor takes on mCloud

Scene: Smokers zone, 9 am, office backyard.

"Hey, Victor, good morning!"

"Hey, Smith, good morning!"

"What's up these days? Oh man, look at these. You've have already piled up heaps of cigarette butts all around. Are you game for a Guinness record anytime soon?" said Smith, with his usual smile.

"It's the regular project frustration which I'm sharing with these friends of mine until they provide me with an answer," winked Victor.

"Seems like your friends are increasing with each passing day! What's so frustrating in the first place?"

"What else could it be other than designing tests for the mobile app project? I designed, executed, and was about to send a report on the tests run on a mobile OS for our project, which is going to be released in a fortnight."

"And what happened?" interrupted Smith.

"These mobile development platforms have entered into an unending battle, bringing in newer versions like crazy. All my efforts are now in vain. To spice it up, this company on whose devices we tested has got in a new model just last week and the market is proving that it would replace every existing device in a short space of time," vented Victor.

"So what's the deal? Can't you post a disclaimer stating that your app would work on the tested configurations?" questioned Smith.

"Come on, Smith, you've tested mobile apps for years. Don't you understand how a customer thinks? He expects your app to work on his device, rather than having to keep his old device for the sake of your app," replied Victor sharply.

"Oh yes, I do. Don't you think it should be possible for you to execute those tests again on the newer device? After all, you are working on Agile. You shouldn't be taking more than a fortnight to thoroughly test again on this newer version. That's the same with the test cases in hand, this should work like a charm."

"What makes you think I would be here if I had the device, rather than testing it. In addition to the platform battle, this is exactly what has complicated the situation. We don't have a physical device to hand. How can we be sure that our app would work on this platform without even having looked at it at least once? A built-in emulator is a solution. But they cannot guarantee that it would work on a physical device in the same manner. There are other variables like network, interruption, memory usage, etc, in play on a real device and these are the riskiest bet you can go with by just checking on an emulator."

Victor continued: "And what makes it even more complex is that these devices are not yet available in our market. We need to wait until they arrive and the purchase team has made it clear that they would be available at best in a month and not before that. Management is not happy about buying a device every month in every development location to test on it only for few months and then find that the device is now obsolete and is lying idle or only used to download game apps," Victor paused to inhale the smoke deeply and transform it into an attention-grabbing *inside-out smoke ring*.

"Wow! You've started to make tricks from the smoke, impressive!" continued Smith, "But can't you take a cue from these clouds of yours? These have already shown you a solution, why do you still want to bother with them? Can you please let them go, for goodness sake? This passive smoking is choking me."

"What do you mean, they have shown me a solution with the ring? Do you want me to make a living by displaying tricks with smoke clouds?" chuckled Victor.

"I mean, why don't you start using cloud for your testing rather than messing about with these clouds?" answered Smith, wittily.

"Oh no! No more puzzles for today. My brain is already overloaded."

"OK, let me put it simply. You have these problems:

1. Myriads of OS and devices are springing up each day.
2. You don't have a real device to hand to test your app.
3. These physical devices at best last only for a few months.
4. Management is not seeing a return on the investment made in purchasing devices
5. You need to test on a real device to gain confidence.
6. The development team is spread across the globe, and each one of them needs a device.

"The solutions for testing your app that you have been using up to now are either an emulator or a real device. You have totally forgotten another layer which could come in between."

"What's that?" said Victor, sharpening his ears to hear the magic word.

"It's cloud or mobile cloud testing. Let me explain to you how it works:

1. The cloud company which provides you Software as a Service (SaaS) or Infrastructure as a Service (IaaS) has a suite of thousands of every possible real device connected on various network, most often in different countries.
2. You contract with them for a plan to pay either per use or at a certain frequency and the device is set to you.
3. You connect to the device through the cloud, deploy your app on it and test it like a physical device.
4. Since it is a real device with a real network, you are close to testing the actual behavior of your app on the device.
5. You may choose from the pool of devices to suit the combination of manufacturer, platform, operating system version, etc.
6. Since you are going to complete your testing in few days, you release back the device and save on buying the physical devices and, importantly, setting them up.
7. The best part is, since it is cloud-based, your team can access the device from any part of the world."

"Smith, this is timely information. I am delighted I got a solution to all my problems."

"Oh, oh...Hold on my friend, I haven't finished yet.

Apart from functional testing, performance testing with cloud is much simpler. So, if you want to simulate a load rather than setting up huge infrastructures, use the power of cloud to use it on an as-needed basis. Load generators are with you and millions of users could be instantly simulated. But..."

"No, no buts please..."

"Well, this 'but' is to caution you on the usage.

1. In spite of being real devices, these might be sluggish sometimes.
2. Pay special attention to the costs. It might be cheaper to buy a device rather than using cloud if the project is going to continue for a longer period.
3. Since it is a layer in between emulators and real devices, it can only complement the other two and can never replace them.
4. The best strategy is to have emulators at the start, then switch to cloud testing, and in parallel have a real device to test. Bring down the usage on cloud after you get a real device, since you are charged for every minute of your usage if you choose such a plan."

"Hmmm...These issues are important. I will consider them."

"Major vendors are competing fiercely in the market. Evaluate them with the free trial to check if the solution suits you."

"Definitely, Smith. Thanks a million. My current problems seem to have vanished like those smoke rings."

"I'm pleased I could help you, Victor. However, you still seem to want your cloud friends when a real friend is right in front of you. They cost you per usage," smiled Smith, to find Victor reciprocating the smile and dropping his cigarette. ■

> about the author

Mithun Kumar S R works with Schneider Electric on its building automation products. He previously worked with Tech Mahindra and Siemens Information Systems Limited on a global telecoms project and magnetic resonance imaging scanners respectively.
Mithun is an ISTQB Full Advanced Level certified professional, and coaches certification aspirants. He regularly writes articles in international magazines and speaks at conferences. He is also a LONMark Certified Professional and is BrainBench certified in Software Testing and Software Quality Assurance. He holds a bachelor's degree in Mechanical Engineering and a master's in Business Laws.

By Rajesh Mathur

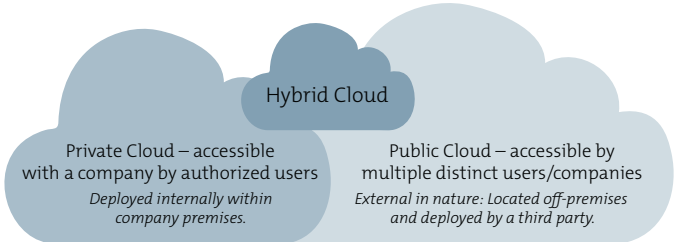
Testing in the Cloud: Bane or Boon?

In the last decade there have been many technological advances. In today's world, where the global economic crisis has affected the operations of many major corporations and has even forced many technological groups to either reduce their operations or workforce extensively, the focus has constantly been moving towards cost savings measures. Companies are forced to look for newer and better ways of working and the CIO's are forcing their teams to look for innovative solutions that help them deliver similar or better results with significant cost savings.

Software testing has always been seen as a necessary evil by most of the organizations. Test managers can no longer consider hiring a large number of resources to deliver solutions, or buy expensive tools, or spend millions on building a test environment. In such a scenario, when the Cloud started appearing in the technology sky, everyone was thrilled. And most thrilled were the senior management who saw it as a single solution for all their infrastructure needs and problems. Now, it is true that the Cloud helps save money, but it is definitely not a one-size-fits-all solution. So, when the Cloud started being discussed among senior management in companies across the world, testing infrastructure was also being mentioned and was seen as a costly affair. Before examining further whether or not Cloud computing can help software testing, it is important to understand exactly what Cloud computing is.

In its most basic definition, Cloud computing is an Internet-based computing infrastructure, which extensively uses shared software and hardware resources and provides these on demand as a service. Since Cloud is provided as a service, it obviously reduces the efforts of client organization. Clients benefit in the form of rapid time to market, more convenient scalability, and even a reduction in infrastructure requirements, and the strong opportunity to reduce capital expenditure significantly.

There are many large technology companies that provide Cloud services to other organizations. Some of the well known names are Microsoft, VMWare, Amazon WebServices, and Citrix. Many of these companies offer services either as Platform as a Service (PaaS) or Infrastructure as a Service (IaaS). Basically, the Cloud deployment is provided either as a public cloud or a private cloud. In certain cases it is also provided as a hybrid cloud which is a hybrid of Public and private clouds.



When it comes to software testing and Cloud computing, there appears to be some controversy about the understanding as well as of terminology. Almost four years ago, one of my supervisors asked me why we were not focusing on Cloud testing. I was commanded to find out whether Cloud testing was feasible or not. I was sure that at least I did not have enough capability to test a cloud. However, Cloud testing is in fact a misnomer for testing in the Cloud. So, it is important to analyze the claims that testing in the Cloud revolutionizes the way testing is performed. While it is possible to claim that Cloud brings cost benefits, my view is that it does not directly affect the delivery or approach of functional testing. Non-functional testing, such as performance testing, does benefit from testing in the Cloud.

There are several reasons why testing in the Cloud should be accepted by companies. Testing is now being accepted early in the SDLC and most companies understand the value of introducing testing early on in the life cycle. The infrastructure requirements for various testing levels begin with the unit testing phase and continue with the system and integration phases, user acceptance phase and non-functional test phases where separate platforms might be required for performance, security, recoverability/resilience/disaster recovery or operational acceptance testing. While many companies attempt to reduce this cost, most of the time it is not possible to provide only one test environment for all of these testing needs. Availability of one or only a limited test environment has a significant effect not only on the delivery schedule but also on the time to market. It is also evident that the more people resources have to wait for a test environment to become available, the more frustration levels increase, along with the possibility of error. A Cloud solution can easily resolve this situation. A company which considers renting a cloud for its testing needs, can more easily request private clouds within the company and make these available to different teams or departments to resolve risk of parallel environment use or interdependence among various testing phases. This option is much more cost effective than investing an enormous amount on testing environments and then letting them rot in the basement after project delivery. Availability of a testing cloud facilitates convenient access to a network of a shared pool of hardware or software resources which can be configured according to demand.

However, we should not forget that Cloud does not basically change the approach of testing delivery. We have only changed the model of the test environment or infrastructure delivery. Those who argue that Cloud computing actually changes or has changed the way testing is performed possibly have made the wrong assumption regarding testing as well as about the Cloud's capabilities for software testing.

Firstly, it is claimed that clouds are mostly secure and test data can be conveniently used across test phases in a Cloud environment. In actual fact, testing on a cloud requires much more stringent standards and checks on test data, only because a cloud always has the possibility of a security breach, especially in a public cloud. It is not an uncommon practice that data from production environments is taken to be used as test data after obfuscation. What if, due to human error, production data is exposed in a public cloud and there is a security breach? There was a well known recent case of hacking on a major Cloud service provider.

Then there are concerns about transitioning legacy systems on the Cloud. While virtualizing some of these systems might be possible through interfacing with the client's existing networks and/or infrastructure, it might still require additional effort and infrastructure at the client's end, thus reducing the benefit profile for testing deliveries.

While there are strong arguments on both sides of the case, my personal belief is that testing on the Cloud will emerge more strongly in the coming years and demand for clouds will make them more affordable and convenient even for smaller organizations. Cloud definitely breaks down the constraints of resource availability within organizations and lets testers focus on real testing rather than forcing them to focus on issues related to the testing environment. ■

> about the author

Rajesh Mathur is the Test Delivery Manager at Hong Kong's premium airline Cathay Pacific Airways. He manages testing of multiple projects under Airline Operations and Cargo domain. With over 16 years of experience in software testing, Rajesh has lived and established his career in countries like US, UK, India & Hong Kong and has worked as tester, test lead, manager, Programme Manager and Test Delivery Manager for many high profile multi-million dollar projects.
Through his career, Rajesh has been active on the testing arena for a long time and has been actively working on strengthening the testing community and the practice of software testing through trainings, mentoring and by participating in conferences.
Rajesh holds degrees, diplomas and certificates including BS with Physics & Maths, Master of Arts with specialization in Statistics, Post graduate Diploma in Computer Applications, MCSD, ITIL, BBST, ISEB etc. Rajesh is a member of Australian Computer Society, Hong Kong Computer Society and a former Member of Computer Society of India.

learntesting

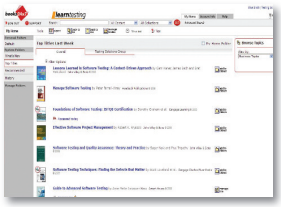
The Testing Knowledge Library

Access to over €2,500 Value Books for just €25!

Testing Experience is delighted to offer readers of the magazine a 50% discount on the annual subscription to the Learntesting 'Testing Knowledge Library'

Visit www.shop-learntesting.com and use coupon code TL50EUR (or replace EUR with GBP, USD, AUD or NZD)

Learntesting brings you a new service, providing software testers around the world with 24x7 access to an up to date knowledgebase in the field of software testing that is second to none.



- 50 Testing and Related e-books
- Testing Innovation
- Testing Templates
- ISTQB Updates
- Regular Newsletter & Library Updates
- Special Offers

A Global Virtualized Learning Environment

enquiry@learntesting.com
Tel: +44 (0)20 3475 3330

www.learntesting.com

By Khamer Chittanai

Moving Testing to the Cloud – An Exploration

Introduction

Cloud computing recently received significant attention because it changes the way computing and services are delivered to customers. For example, it changes the way computing resources, such as CPUs, databases, and storage systems are provided and managed. Today, leading players such as Amazon, Google, IBM, Microsoft, and Salesforce.com are offering their cloud infrastructure for services.

A recent study has shown that 66% of the companies surveyed are considering cloud-based solutions for the future. Also, 63% of companies in financial services, and 62% in manufacturing are currently using cloud-based solutions. The majority of the companies surveyed stated that they now have streamlined administration, increased agility/scalability, sustainability, better performance, improved security, control of data, and, last but not least, lower costs.

What is Cloud Testing?

Cloud testing uses cloud infrastructure for software testing. It is a subset of software testing in which simulated, real-world Web traffic is used to test cloud-based Web applications. Cloud testing also verifies and validates specific cloud functions, including redundancy and performance scalability.

A number of small to medium-sized IT organizations have migrated to cloud solutions. As a result, cloud testing has become necessary to validate functional systems and business requirements. In addition to cloud experience, cloud testing engineers require knowledge of different types of testing and tools.

Organizations pursuing testing in general, and load, performance testing, and production service monitoring in particular, are challenged by several problems such as limited test budget, and meeting deadlines. High costs per test, large numbers of test cases, little or no reuse of tests, and geographic distribution of users add to the challenges. Moreover, ensuring high quality service delivery and avoiding outages requires testing in the company's own data center, outside the data center, or both. Cloud testing is the solution to all these problems.

Why is Cloud Testing Important?

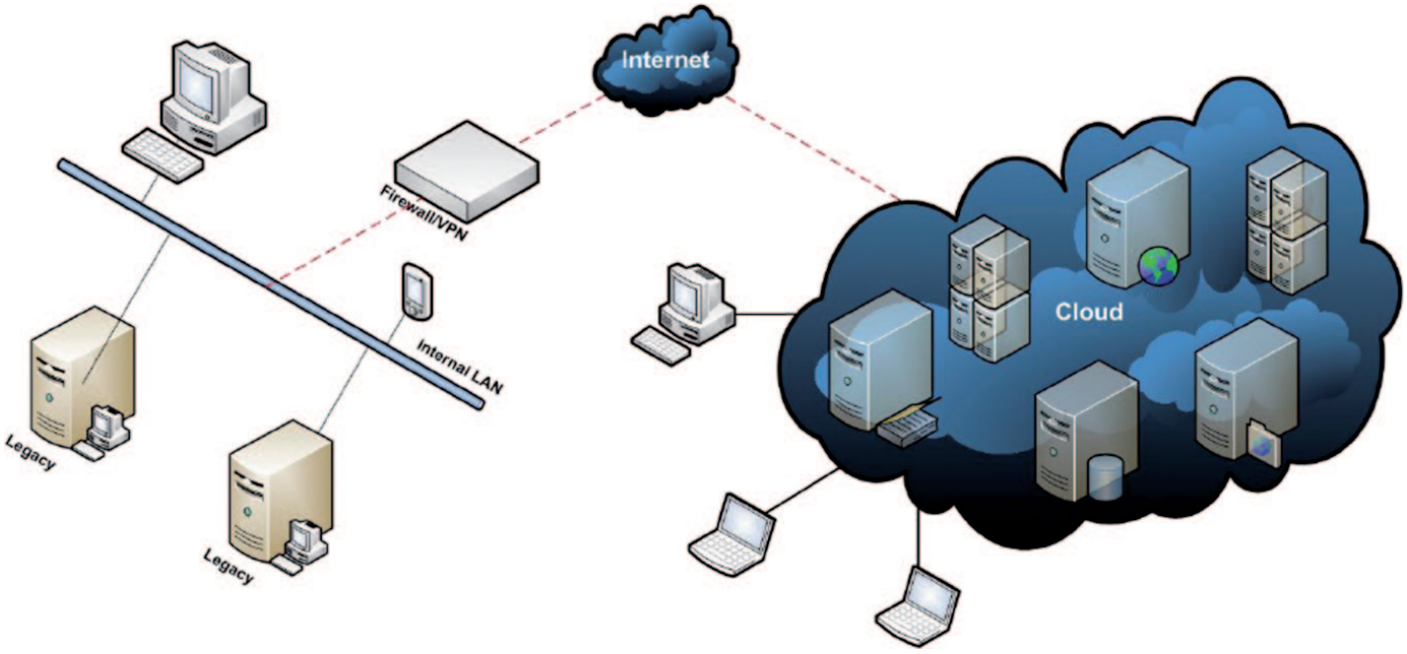
Compared with current software testing, cloud-based testing has several unique advantages which are listed below:

- On-demand test services (by a third party) to conduct large-scale and effective real-time online validation for Internet-based software in clouds.
- Easy to leverage scalable cloud system infrastructure to test and evaluate system (SaaS/cloud/application) performance and scalability.
- Improved product quality and reduction in detected defects by as much as 15–30%.
- Reduction in capital and licensing expenses by as much as 50–75% using virtualized resources.
- Reduction in operating and labor costs by as much as 30–50% by automating development and testing resource provisioning and configuration.
- Shorter development and testing setup time – minutes rather than weeks.

Benefits, Limitations, and Challenges in Moving to Cloud:

Although there are many published papers discussing cloud architectures, technologies, models, design, and management, cloud testing and TaaS are still new subjects in the software testing community. Hence, test engineers and quality assurance managers have encountered many issues and challenges in testing modern clouds and cloud-based applications. Typical questions are listed below:

- What is cloud testing? And what are its special test processes and scope, requirements, and features?
- What types of cloud testing, environments, and forms do we need to perform for SaaSs/clouds and cloud-based applications?



- What are the major differences between conventional software testing and cloud-based software testing?
- What are the special requirements and distinct features of cloud-based software testing?
- What are the special issues, challenges, and needs of cloud testing?
- What are the current practices and tools, and who are the major players?

Using business and technical terms, a few points are listed below indicating the benefits, limitations, and challenges faced by cloud computing. Ironically, the benefits, or strengths, are also sometimes mentioned as weaknesses, but they are relative to the facilities provided by the vendors and the support infrastructure.

Benefits	Limitations	Challenges
▪ Cost reduction ▪ Easy implementation ▪ Speed to value ▪ Flexibility ▪ CapEx avoidance ▪ Greener IT ▪ Highly automated ▪ More mobility ▪ Allows IT to shift focus ▪ Scalability	▪ Data confidentiality and auditability ▪ Service availability ▪ Data transfer bottlenecks ▪ Performance unpredictability ▪ Scalable storage ▪ Bugs in large-scale distributed systems ▪ Scaling quickly ▪ Reputation fate sharing ▪ Software licensing	▪ Manageability ▪ Data governance ▪ Reliability and availability ▪ Virtualization security ▪ Latency ▪ Connectivity ▪ Regulatory compliance ▪ Privacy, legal ▪ Open source, open standards ▪ Sustainability and siting

To gain confidence, create a proof of concept. Ask yourself what cultural and process changes are required to move to cloud-based testing, who owns service management, and what changes in organizational/financial process alignment must be made to manage/provide for a new service.

Getting into Cloud Testing

1. Non-Functional Testing

Cloud computing solutions claim to be scalable on demand. How do businesses verify that the solution delivered is capable of coping with the workload which it is required to undertake? Load or stress testing can be used to prove that the solution developed can be scaled as required. By using test techniques and tools which are capable of applying huge amounts of load on the solution, the cloud can be accurately measured and its capacity verified.

Performance testing techniques allow the system's performance to be measured and verified accurately. Using performance testing and load testing techniques in tandem allows an accurate image to be created of the solution's capability over the cloud.

Security testing can provide assurance that business critical data is being stored and transported safely. Techniques such as penetration testing can prove that the mechanisms which have been developed to maintain security will remain intact during potential attempts to compromise the cloud solution.

2. Functional Testing

How do businesses validate that the system will behave as specified within requirements? System testing techniques allow the system's behavior to be proved within its own entity. Before considering any deployment, it is critical to prove that the system functions as it has been designed, the system components work together, inputs and outputs are as expected, and the overall resulting system is of a suitable quality to release.

Before any deployment, how do businesses verify that the integrated solution will behave as intended to facilitate business continuity? Inte-

gration testing allows the business to verify that the cloud solution will work within the current infrastructure and environments, proving that the implementation of a cloud solution does not detrimentally impact any existing systems.

Finally, the business requirements must be verified and validated to prove that the end result of the cloud solution will meet the documented needs of the business. User acceptance testing will use business requirements to prove that the delivered cloud solution meets those needs.

Looking Forward

Great opportunities to cut costs and move them from capital expenditure into operational expenditure means that adoption of cloud services will continue to increase. To that extent, serious engagement is not optional – it is imperative because the opportunity is enormous. Looking forward, we anticipate an incremental take up of testing on the cloud and this is explained below.

Near Term: Cloud solutions, and, consequently, testing, will gather momentum, and offerings will gain maturity. Organizations will increasingly implement their development and test environments in the cloud, and performance testing in the cloud will become more routine through a managed test line model. Increasingly, activity will focus on building private, hybrid, and public (test) clouds and testing cloud-based applications for a growing number of adopters. This will create the need for more services based on assessment, implementation, and strategy.

Long Term: Cloud will become a full model for computing, but as an additional platform rather than a replacement for mainframe and client/server. Service providers will develop a broader base of offerings from the test cloud business models, such as creating cloud-based test environments, testing on cloud-based environments, testing cloud-based applications, using SaaS for testing tools, and becoming a broker for test cloud services. There will be a corresponding decline in demand for infrastructure/data center services.

Conclusions

Cloud computing has received significant attention recently because it changes the way computing and services are delivered to customers. A number of small to medium-sized IT organizations have migrated to cloud solutions. As a result, cloud testing has become necessary to validate functional system and business requirements. In addition to cloud experience, cloud testing engineers require knowledge of different types of testing and tools.

There are many benefits in moving to cloud testing, but there are also many risks. This paper has covered the areas organizations should bear in mind as they consider whether the benefits of moving testing to the cloud are worthwhile.

References

[1] L. Ciortea, et al, 2010. Cloud9: a software testing service. *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4.

[2] Vengattaraman, T., Dhavachelvan, P. and Baskaran, R., 2010. "A Model of Cloud Based Application Environment for Software Testing", *International Journal of Computer Science and Information Security (IJCSIS)*, Vol. 7, No. 3, 2010.

[3] *http://highscalability.com/blog/2010/5/26/end-to-end-performance-study-of-cloud-services.html*

■

> about the author

Khamer Chittanai has over 12 years of experience in the field of testing and in various domains. He currently works as a project manager in Honeywell Technology Solution, Bangalore and has been part of the Testing CoE for the past 10 years.
He is also involved as internal faculty, delivering engineering process training courses at the organizational level and is the functional area representative for the Testing Group in CMMi Level5 Certification Assessment. He has over 300 hours of training and mentoring experience.
Khamer holds a bachelor's degree in Mechanical Engineering from Bangalore University and is also an ISTQB Certified Tester Advanced Level.

Prof van Testing recommends

Díaz Hilterscheid

twitter

Follow me @vanTesting

IREB – Certified Professional for Requirements Engineering – Foundation Level

Description

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

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

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

IREB International Requirements Engineering Board

Dates*	3 days
Mar 19–21, 2013	Oslo, Norway (EN)
Mar 19–21, 2013	Berlin, Germany (DE)
Apr 9–11, 2013	Mödling, Austria (DE)
May 14–16, 2013	Berlin, Germany (DE)
May 14–16, 2013	Helsinki, Finland (EN)
Jun 18–20, 2013	Berlin, Germany (DE)

* Subject to change.

Website:
training.diazhilterscheid.com

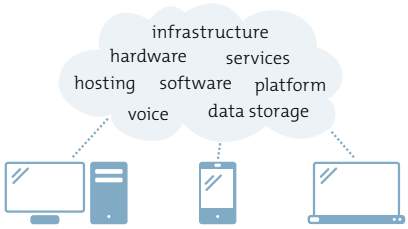
By Narayana Maruvada

Testing in the Cloud

This article is primarily intended to provide a quick overview and portray the importance of the Cloud computing business model followed by a detailed insight into two of its sub-sets and areas of prime importance i.e. Cloud-based testing and testing a Cloud.

Cloud Computing – A New Orientation for Carrying out Business

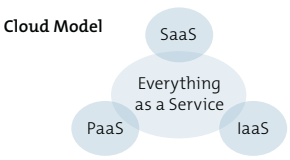
In recent times, organizations have become very curious about this term 'Cloud computing' and seriously trying to understand and identify the various possibilities available and likewise planning strategically to use and/or implement it to reap its benefits. Although this term is not new and has been in existence since the very inception of any formal business, it is now essential for us to understand and appreciate the various improvements that are likely to be happening in it and its allied areas.



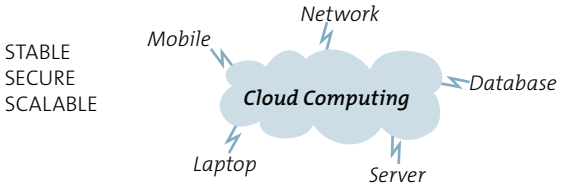
In general terms, Cloud computing is understood and appreciated as a basic formal business model, which has been deployed and executed for various material needs. However, in the current scenario, this very term has been made more formalized in the context of IT products and services. So taking this very IT context into consideration and talking about it strictly from a novice's perspective, it can simply be just described as 'a service' or 'on demand'. In order to offer this wide range of products and services on demand, Cloud computing has been classified into the following major categories on the *basis of delivery*, namely:

- **Software as a Service** (SaaS) – where a variety of applications and products are available on demand for any internet user.
- **Platform as a Service** (PaaS) – where a variety of run-time environments is available on demand and used by the developer community to develop and deliver their applications.
- **Infrastructure as a Service** (IaaS) – where various computing resources such as storage, networking components, middleware etc. are available on demand and can be accessed or made use on a 'pay per use' basis.

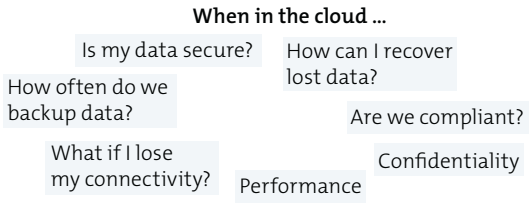
Similarly, taking the *deployments* factor into consideration, Cloud computing is in-turn categorized into public, private, community, and hybrid Clouds etc. Now, irrespective of the aforementioned classifications, Cloud computing is intended to provide the following benefits:



1. The cloud business model is very scalable, i.e. Cloud-based applications and/or products can be customized easily based on the requirements.
2. The service or product that is offered through Cloud is available as and when it is required, i.e. it is possible to obtain access to the required resources seamlessly.



But, on the other hand, organizations still have questions about some of the challenges involved in adopting the Cloud computing business model, which are mainly concerned with data integrity, security, availability, and privacy etc. However, even with these major limitations and/or challenges the Cloud computing business model is indeed advantageous to organizations, provided these risks and/or challenges are mitigated by implementing appropriate verification and validation techniques. Now, this introduces us to two new terms: **Cloud-based Testing** and **Testing in a Cloud**.



Cloud-based Testing – Redefining the Perspectives of Testing

It is the new vista of opportunity for testing that was both envisioned and opened up by the Cloud computing business model. But, from the IT organization's standpoint, testing has always been viewed as a challenge because of the sporadic infrastructure and resources that it demands. Additionally, the growth in complex business applications is making it very difficult for organizations to augment the testing facilities they require to simulate the real-time environment. However, thankfully technological virtualization has arrived to meet the organization's base level operational and financial objectives by eliminating the need for huge capital investments. But, even after providing the requisite set-up costs, many organizations are failing to achieve operational flexibility and scalability. So, Cloud-based testing arrives with the requisite potential to mitigate these operational and scalability related issues by offering a very compelling combination of lower costs, pay-per-use, and elimination of overheads. As a result, Cloud-based testing indirectly enables organizations to both experience and benefit from **non-cost** factors, such as:

- Achieving greater levels of efficiency.
- Better collaboration.
- Hassle free approach for holding and maintaining assets.
- On-demand flexibility and, importantly, reduced *time to market* for key business applications.

Shifting Testing to the Cloud – A Highly Optimistic Move

As stated earlier, although organizations are moving in favor of Cloud-based computing with the utmost optimism and expectations, but testing alone happens to be one area which appears very adventurous and exciting while shifting it to the cloud because of following reasons, namely:

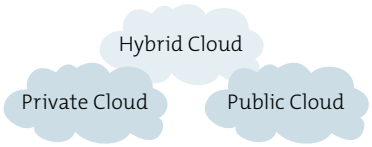
- Testing is cyclic and requires the environments to be set and updated based on the project requirements.
- Although testing is considered to be important, it is felt to be a non-critical activity. So, moving testing to the Cloud is a safe move because it has minimal impact on the ongoing business activities.
- Importantly, applications are becoming increasingly complex and dynamic, which, in turn, introduces a multiplicity of challenges for testing teams. For example, Web-based applications have to be tested in multiple operating systems and their updates, likewise, on multiple browser platforms and their versions, different type of hardware, and with large numbers of concurrent users to understand the application's performance in real time. So the conventional approach of establishing test environments for testing such complex applications will incur huge capital and resource expenditure.

It is here that *Cloud-based Testing* comes to the rescue to leverage the requisite infrastructure there by, reduce the unit cost incurred in computing, and importantly both improve and increase the effectiveness of the test.

Operational Benefits of Cloud-Based Testing – Promising and Quantified

→ In the current scenario, cost reduction is the key factor influencing organizations to embrace Cloud-based testing. The same has been inferred from the majority of surveys which outlines that economic benefit is the key driving force behind the adoption of Cloud-based testing by organizations. However, the magnitude of all these benefits depends on various factors and prominent among them is the *type of cloud* chosen. So, it is equally important to understand and appreciate the various types of Cloud model's available. Basically, there are three types of cloud environments that are available:

- Public or Fully Managed Cloud – this encourages organizations to shift to a flexible operating expenditure model, since they need not concentrate on upfront investments in areas such as infrastructure, tool licenses, configuration, maintenance of test environments, etc. The advantage remains that the organization only pays for what it uses.
- Private Cloud – this option ensures that the supporting infrastructure can be decommissioned once the testing process is complete, which, in turn, frees the organization from incurring any operational costs. As a result, it provides the organization with benefits such as customization, efficiency, availability, resilience, security, and privacy.
- Hybrid Cloud – this option encompasses the merits associated with both public and private Cloud.



→ Similarly, other two factors which may equally judge the benefits apart from the type of cloud include *configuration of the test environment* and the *type of tests* that are conducted. Likewise, additional factors such as the ability of the testing team to choose the right Cloud service provider and collaborate with them for the optimum utilization of cloud resources also play a crucial role.

→ Furthermore, leaving aside the cost issues, organization's next immediate concern is to minimize the percentage of defects that generally occur due to inexact configuration of test environments. Fortunately, even this is well addressed by Cloud-based testing, as it offers a standardized infrastructure and other supporting pre-configured items which tend to reduce the defects considerably. Organizations can achieve this infrastructure standardization by making use of a catalogue which delivers the required discipline and commitment to meet the service level agreements. Collectively, all these disciplined activities lead to a better provisioning of the test environments which in-turn helps organizations meet their operational objectives.

→ On similar lines, *time to market* happens to be another big concern for organizations as it is often restricted by conventional test environments because it involves creating in-house test environments and this can be time-consuming. Unlike, Cloud-based testing facilitates the *on-demand provisioning* option. Using this option, the requisite testing resources which are already in the cloud can be provisioned immediately to meet the time-to-market demands.

→ Importantly, organizations are relieved of the concerns regarding finding and augmenting high-end servers, procuring licenses for all the supporting software and testing tools, and getting them installed on the machines. With Cloud-based testing, the service providers provide the testers with requisite access to the ready-to-use virtual labs which encompass the entire gamut of system softwares, test management and execution tools, supporting middleware, and storage needed to simulate a test environment that closely resembles the real-time environment. So, by using these virtual labs, testers have the utmost flexibility to run the applications with minimum effort and at maximum ease merely by utilizing the pools of virtualized infrastructure to scale up the test environments based on the type of testing and test requirements. Similarly, Cloud-based test environments enable the testing teams to have greater control over building and executing tests, analyzing application performance, and looking for bottlenecks and stress areas, by allowing them to scale millions of concurrent users, and assess the breaking points and capacity thresholds to handle highly unpredictable application requests.

So, such assessments provide testers with a clear picture of possible errors and down-times, etc., on the basis of which they can mitigate them with proactive measures. Above all, the built-in collaboration and management tools which are available with Cloud-based testing facilitate realistic collaboration between dispersed teams, which, in-turn reduces the delay in back-and-forth communication between teams.

Operational Challenges with Cloud-based Testing – a Thoughtful Consideration

Although Cloud-based testing has its own clear benefits to offer, organizations should prepare to contend initially while dealing with the following challenges before they reap the stated benefits. Those challenges are:

- Lack of standards – Currently, there are no universal standards and/or guidelines available, which can be used to integrate the Cloud resources with the user company's internal resources. There might be integration issues because the public Cloud providers have their own architecture and operating models etc.
- Interoperability issues – These generally occur as a result of incompatibility between in-house resources and Cloud-based resources, and are also due to restricted access or implementation.
- Security in the public Cloud – This is another big concern, since the current encryption techniques may not be sustainable and might be considerably inefficient.
- Terms and conditions that are imposed by the Cloud-based testing service provider must be thoroughly understood and evaluated by the organization and then a proactive decision has to be taken accordingly.
- Proper planning – An organization should have a clear road-map and, similarly teams should rigorously plan their test environments and other supporting infrastructure from utilization through to decommissioning. Importantly, an organization should keep track of the expenses incurred while using the Cloud services.
- Performance constraints – If organizations are opting for the public Cloud, there might be cases where performance issues are encountered since public Clouds are shared by a number of users.
- Infrastructure – It has to be thoroughly ensured that what is supplied should be good enough to simulate the real-time environments.

As mentioned earlier, Cloud-based testing has also introduced a new set of challenges such as *data integrity*, *security* and most importantly, lack of *well-defined standards* to carry out business in the Cloud, in addition to the existing operational and technical challenges. So, all these collectively have provoked organizations to examine themselves closely and explore the various ways they can defend themselves from exposure to any of the existing or newer challenges. So, collectively all this introspection and thought process has given rise to a new area of prime importance called *Testing a Cloud*.

Testing a Cloud – a New Variant of Testing, Conceptualized

This technically refers to verification and validation of the applications, products, environments, and infrastructure that are available on demand, to ensure that all of these conform to the standards set by the Cloud computing business model. As part of testing a Cloud, an organization should go with using both conventional and contemporary verification strategies. Irrespective of whether a product or service is offered in-house or on the Cloud, they should meet the functional requirements. Likewise, equal emphasis is required on non-functional requirements for Cloud offerings. So, the various types of testing that can be carried on the Cloud are:

Availability Testing – this is intended to ensure that the Cloud offerings are available at all times and there are no abrupt down-times.

Disaster Recovery Testing – this is intended to ensure that the Cloud services are back online with minimum adverse effects on the business.

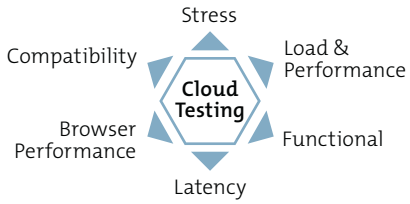
Interoperability Testing – this is intended to ensure that any application which is either developed or available on the cloud works in multiple environments and platforms.

Multi-Tenancy Testing – This refers to the scenario where multiple clients and organizations use the Cloud services on demand because requirements vary from one organization/client to the other, the Cloud service offerings should be customized accordingly and should provide data and configuration-level security. So, here the Cloud services offering should be thoroughly validated for each organization or client.

Performance Testing – this is intended to ensure that the performance of the application(s) remains intact, even with maximum requests. Generally, performance testing for a Cloud service is different from that for in-house services, and the major objective is to test the Cloud services against fluctuating usage. Even for the Cloud, the following two conventional performance testing methods are used:

- Stress testing
- Load testing

Security Testing – this is intended to ensure that the shared data integrity is maintained and secured at all times. Importantly, there should be no scope for unauthorized access to data.



These are the most important types of testing that revolve around Cloud-based applications. Nevertheless, to decide on the type of verification and validation strategies to be applied, the Cloud vendor should have a thorough understanding of the technical and commercial aspects of the Cloud offerings. ■

> about the author

Narayana Maruvada is a computer science and engineering graduate, currently working as Senior QA Engineer with ValueLabs. Narayana has more than 6 years of experience working on both developing and testing web-based applications. Major area of work and expertise in testing the applications and products that are built on an open-source technology stack for different domains. Keen and inclined for assessing the system's functional & non-functional attributes, investigating and implementing new testing tools, techniques and methodologies followed by contribution for augmenting best practices and quality standards for process improvement.

Erik van Veenendaal

PRISMA: Product Risk Assessment for Agile projects

Column

Risk assessment and management is the backbone of sequential development models but how does it fit in agile environments? How can we be sure to identify new risks when they emerge and to ensure our understanding of all risks remains accurate? In agile much emphasis is on communication. Perfect for development issues where mistakes can be discussed and fixed. But product risk are not by nature iterative: they is absolute and exists all the time and making mistakes in dealing with them may not acceptable. Hence the discussion and consensus approach needs to be slightly formalized by the use of a systematic method and process. That's where PRISMA comes in.

PRISMA

PRISMA (Product RISk Management) is an approach for identifying the areas that are most important to test, i.e., identifying the areas that have the highest level of business and/or technical risk. The PRISMA method has been bottom-up developed by Improve Quality Services in practice over a large number of years. PRISMA has been proven to be successful in supporting (test) organizations as they apply risk-based testing. Today, it is taught at several universities to IT students. The PRISMA approach especially supports the test professional in performing product risk identification and product risk analysis as well as in working in close co-operation with stakeholders.

Product Risk Matrix

The central theme in the PRISMA process is the creation of the so-called product risk matrix (see figure 1). For each product risk identified, the impact of possible defects and the likelihood of these defects occurring is determined. By assigning numeric values to both impact and likelihood, a product risk (test item) can be positioned in the product risk matrix. The standard risk matrix is divided in four areas each representing a different level and type of risk. A different level and/or type of risk should also imply a different test approach, to be documented in a (master) test plan. The product risk matrix can thus used as a basis for all testing performed in a project.

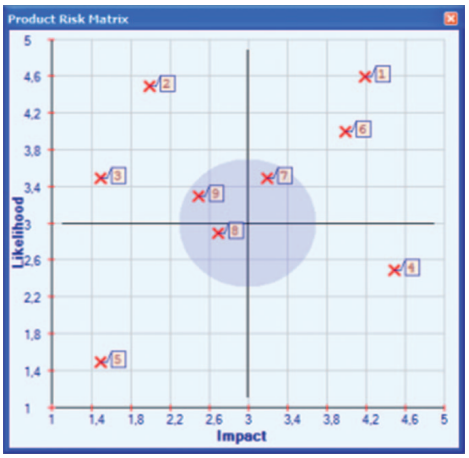


Figure 1. PRISMA product risk matrix

A picture is often worth more than a thousand words. Presenting risk assessment results in a diagram is usually much more effective than in tabular form with many numbers. The table becomes indecipherable very quickly, and often stakeholders lose themselves in a number based discussion. Presenting the results of a risk analysis in a matrix format, as in a PRISMA product risk where impact is on the horizontal axis, likelihood is on the vertical axis, and the four quadrants each represent a level and type of risk – generally provides a much better basis for discussing and validating the product risks.

Agile

Since risk mitigation is one of main objectives of Agile, an approach such as PRISMA can fit into an Agile development project perfectly. In practice PRISMA has proven to be a relatively light weight approach (unlike some), focused on producing tangible results, e.g., the product risk matrix and a differentiated risk-based test approach. Most often when organizations come from a more traditional environment using a structured testing approach such as TMap many testing practices are removed from day-to-day practice. One of the testing practices that is still necessary is a product risk assessment which determines where and how to focus the limited test resources to effectively meet the project deadlines. Where some methods use very detailed approaches for product risk assessment, PRISMA is generally considered relatively light weight and result-driven. In fact, from personal experience, most projects that convert to Agile software development keep PRISMA as one of their core testing practices. Note that in Agile the team is explicitly responsible for the quality of the product.

The risk assessment process

How is PRISMA applied in Agile software development? Product risks are derived from documents (i.e., the list of backlog items assigned to the next sprint and user stories) and are typically identified in a brainstorm session(s). Of course the approach largely depends on the Agile approach that is being used and the cycle time. Based on personal experience, longer sprints of four weeks or one month are most common. The sprint team is often also the PRISMA team performing the product risk assessment. "External" stakeholders are contacted and asked for their input or actively participate in the process. It is usually carried out as a focused meeting, where the team runs through the PRISMA process as described below. At the end of the meeting the team agrees on the product risk matrix and thus the focus of testing.

Risk poker

Having the list of product risk, they are now scored (separately for likelihood and impact) using the essentials of the planning poker technique as often practiced in agile projects. Planning Poker is a consensus-based technique for estimating. It is a variation of the Wideband Delphi method. The PRISMA risk poker is uses the list of product risks (user stories) to be tested and several copies of a deck of cards. The decks have numbered cards and often use the sequence: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, and optionally a "?" (unsure) and a coffee cup (I need a break). A common variation is not using a deck with numbers but colored cards, e.g., dark green, light green, yellow, orange and red, relating back to the "1 to 5" value set. This is practiced since the meaning of the numbers from the deck often lead to much discussion, and are ambiguous in the PRISMA context, when using them to estimate likelihood and impact.

Each team member receives a deck of cards with varying values (or colors). After a short explanation of the product risk item (user story), the moderator (e.g., a SCRUM Master) calls for an estimate for either likelihood or impact. After a few seconds of contemplation, each team member selects a card, without showing it to the other team members, and at a set time, all show their selected cards. It is important that all cards are shown at once, to prevent 'peer pressure' towards a lower or higher number (or color). If the numbers (or colors) are essentially the same, the moderator writes down the median value. If they differ wildly, the lowest estimator and highest estimator briefly explain their choice essentially going back to the PRISMA factors for likelihood and impact. Often then agreement is achieved for a number (or color) based on that discussion. If no agreement is reached, the moderator, business owner (for impact) or lead developer (for likelihood) act as a tie breaker and chooses a number (or color) from within the range. It is important to move quickly to the next product risk item. Optionally, an egg timer can be used to limit time spent in discussion of each item.

One common variation is providing each team member with a limited number of each value or color, and having them 'use up' each value card in the process. This prevents the tendency of some people to stick to very high or very low scores for all product risks.

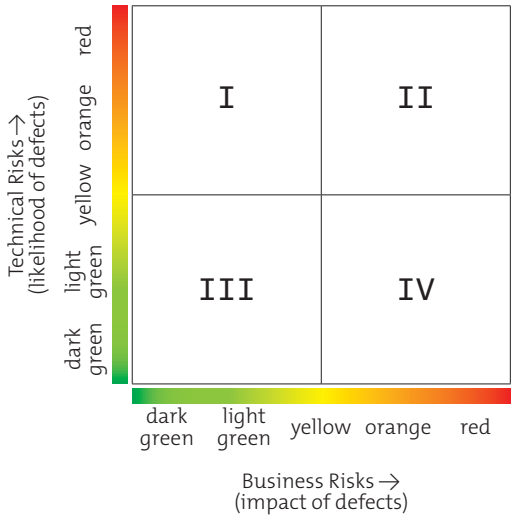
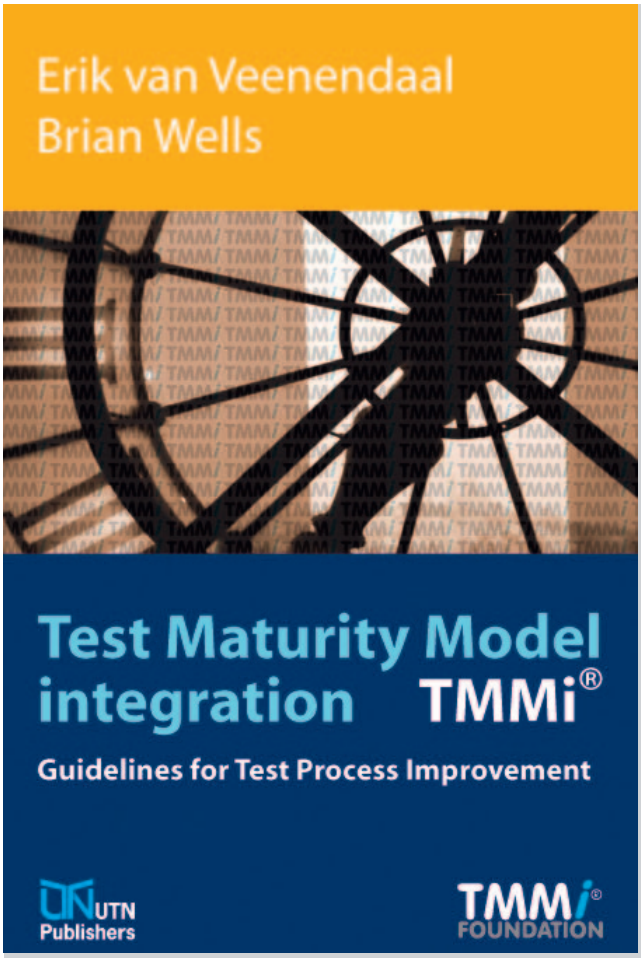


Figure 2. Planning poker Product risk matrix



NEW PUBLICATION

Erik van Veenendaal and Brian Wells

Test Maturity Model integration TMMi – Guidelines for Test Process Improvement
TMMi is a not-for-profit independent test maturity model developed by the TMMi Foundation. The most important differences between TMMi and other test improvement models are independence, compliance with international testing standards, the business-driven (objective-driven) orientation and the complementary relationship with the CMMI framework.

This book provides:
- a comprehensive overview of the TMMi model
- the TMMi specific goals and specific practices
- many examples
- detailed insight into the relationship between TMMi and CMMI.

ISBN 978-94-90986-10-0
pages: 352, price € 39.90
Order at **www.utn.nl**

TMMi® FOUNDATION

At the end of the session when all items have an assigned number (or color) for impact and likelihood, they are positioned in the product risk matrix. The axes of the product risk matrix then usually have a scale that reflects the numbers (or colors) on the deck (see figure 2). The end result of the session (the product risk matrix) is validated by the team to check whether there are no items that are positioned such that they need re-discussion.

One page Test plan

In many Agile projects the product risk matrix including a defined differentiated approach are used as the test plan for the next sprint. By putting a picture (as shown in figures 1 and 2) on the wall, everyone can see the test actions to be performed. This picture is often enhanced by providing the Definition of Done criteria per quadrant. The two to three hour dedicated PRISMA product risk session delivers the sprint test plan in an easily readable format on one page. How much more efficient and effective can one become!? ■

> about the author

Erik van Veenendaal (*www.erikvanveenendaal.nl*) is a leading international consultant and trainer, and a widely recognized expert in the area of software testing and quality management with over 20 years of practical testing experience. He is the founder of Improve Quality Services BV (*www.improveqs.nl*). He holds the EuroSTAR record, winning the best tutorial award three times! In 2007 he received the European Testing Excellence Award for his contribution to the testing profession over the years. He has been working as a test manager and consultant in various domains for more than 20 years. He has written numerous papers and a number of books, including "The Testing Practitioner", "ISTQB Foundations of Software Testing" and "The Little TMMi". Erik is also a former part-time senior lecturer at the Eindhoven University of Technology, vice-president of the International Software Testing Qualifications Board (2005–2009) and currently vice chair of the TMMi Foundation.

By Varsha Jadhav

Testing Applications in the Cloud – a Tactical Approach

Abstract

In the world of technology, a tough economy desires innovation. Any innovation implemented might not eliminate the problems completely, but improvisation definitely helps in the current situation in a variety of ways, such as performance, quality, good ROI, and better business conditions. Businesses in many industries across geographies have used this service model in their business processes, and have achieved great success. This service model results in optimizing resources based on your needs and improved business agility by focusing more on business requirements.

Cloud computing has opened up new opportunities for testing. Testing has traditionally required expensive dedicated infrastructure and resources that were only used sporadically. The growing complexity of business applications was making it difficult for organizations to build and maintain in-house testing facilities that mimic real-time environments. Cloud-based testing offers a combination of lower costs, pay-per-use, a reduction in capital expenditure (Cap-Ex), on-demand access, reduced maintenance, enhanced collaboration, greater levels of efficiency, and, most importantly, short time-to-market for key business applications.

This document focus on Testing in the Cloud as a service model and how organizations can reap all the benefits of cloud-based testing to meet the expected demands of their business by performing comprehensive testing of applications running on a Cloud environment, following the proper approaches to handling the new set of challenges associated with this type of testing, and how to achieve benefits effectively using this model.

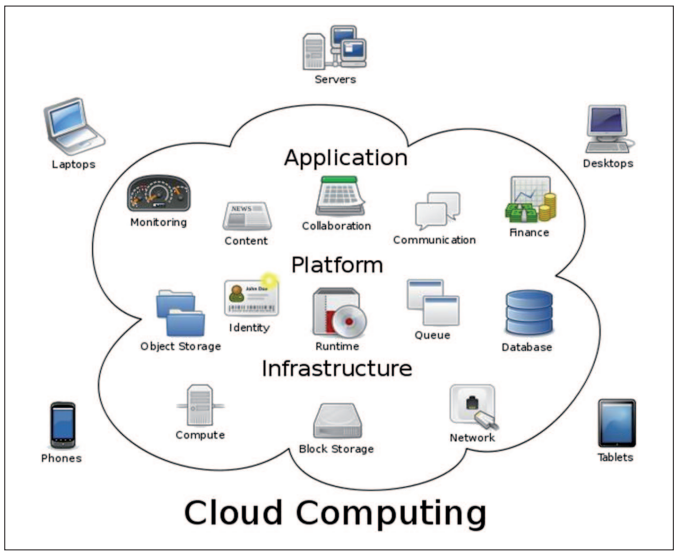


Figure 1.1. Cloud Computing Logical Diagram

What is Cloud Computing?

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. A common misconception among most of us is that *Cloud and SaaS are the same* and the reality is that *all SaaS environments are, by default, inhabitants of the Cloud. However, not all Cloud environments are necessarily SaaS*. Cloud computing provides several categories of service, all offered on demand over the Internet in a pay-as-you-go model. The basic three categories are as shown in Figure 1.2.

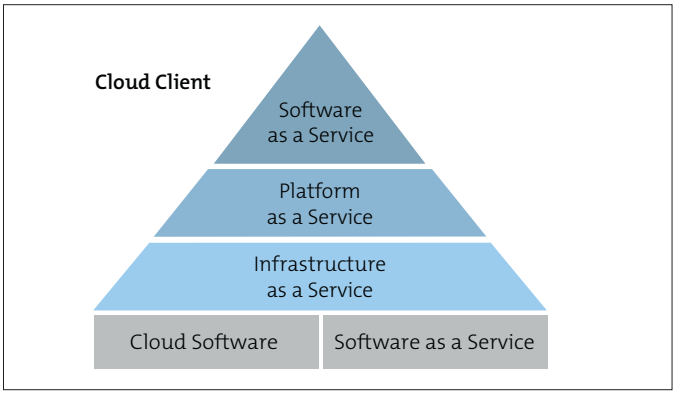


Figure 1.2. Cloud Services

Service Models	Deployment Models
SaaS – Applications and products are available on demand for any Internet user. For example, an online music company charges a certain amount of money for every track downloaded.	**Public Cloud** – Applications, storage and other resources are made available to the generic public by a service provider. Services are offered on a pay-per-use model via the Internet.
PaaS – The runtime environment is available on demand and is used by developers to deliver their applications. The framework for deployment of application code, along with various on demand services, is available as a PaaS offering.	**Community Cloud** – Infrastructure is shared among several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), is managed internally or by a third-party, and is hosted internally or externally.
IaaS – Computing resources such as power, storage, and networking components are available on demand. IaaS is for architects where the actual hardware infrastructure is deployed on a 'pay per use' basis.	**Hybrid Cloud** – A combination of two or more clouds (private, community, or public), offering the benefits of multiple deployment models.
	Private Cloud – Cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party, and hosted internally or externally.

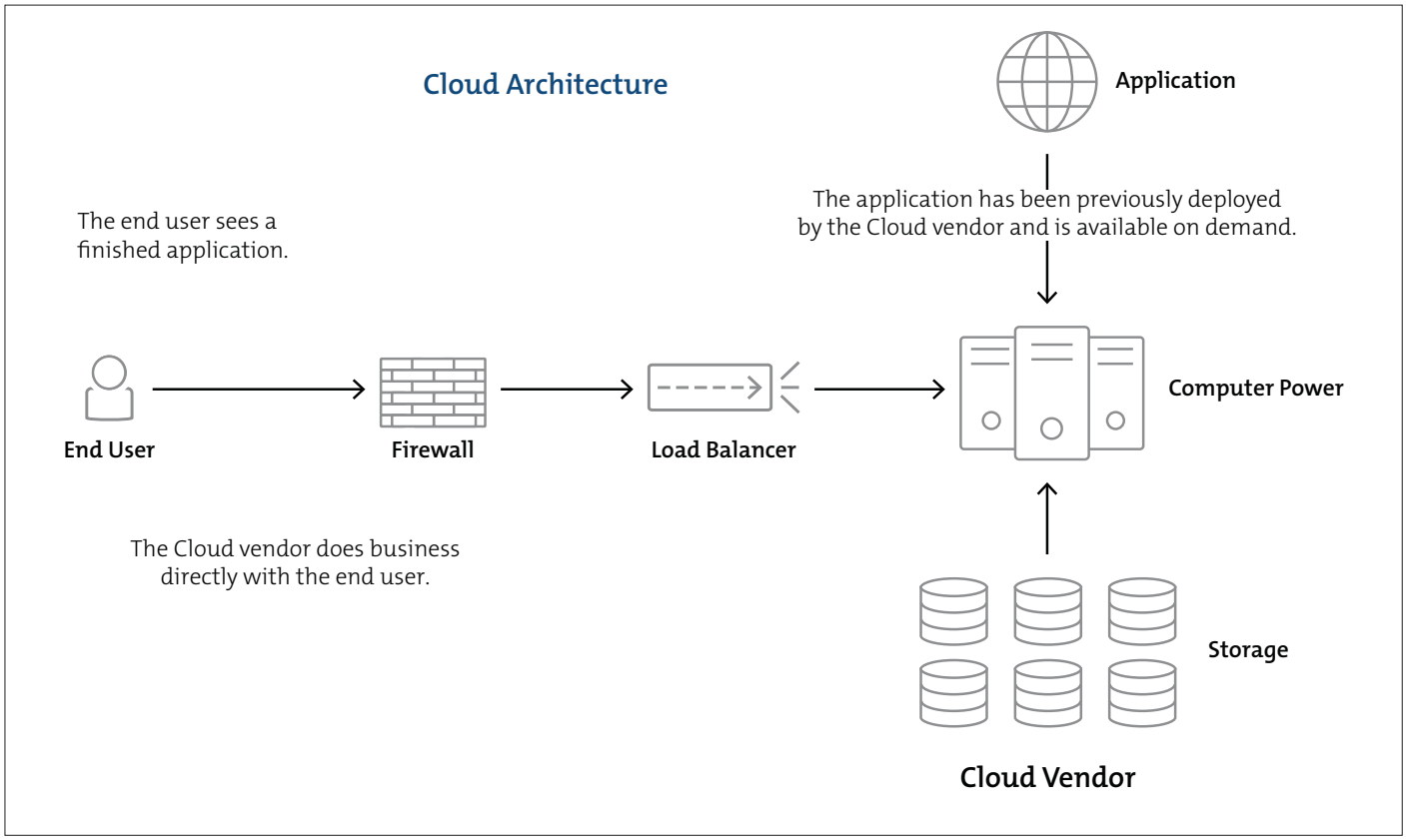


Figure 1.3. Cloud Computing Architecture

What is Cloud Testing?

Testing applications running in a Cloud environment is no different from testing applications running on a non-Cloud environment, but it requires us to merge various types of testing techniques that we all might have worked on in our regular projects. The key areas of focus for testing cloud solutions should be configuration, integration, business, security/accessibility, and performance because the application is being used by a number of customers. However, the appropriate testing methodology can be implemented, depending on business complexity. Seamless integration of Cloud solutions with the customer's upstream and downstream applications is critical to the market success of the product, which is why security testing plays a vital role.

Testing in the Cloud – Why?

- **Low-Cost**
  Cloud provides a cost-effective Internet-based service with low entry, zero infrastructure, and low customization costs.
- **Low Maintenance Costs**
  The consumer is not responsible for the maintenance of applications or servers, because the Cloud vendor does this
- **Mean Time to Market (MTTR)**
  It is just an on-demand service with monthly, quarterly or yearly subscriptions. The service is accessible very quickly within a specific time limit, as you need not purchase the software/hardware and no setups are required.
- **Flexible Contracts**
  In *Cloud Computing*, the contracts are flexible as they are transaction-based, whereas in the traditional model they tend to be fixed.

How is Testing in the Cloud Different from Traditional Testing?

- Complete utility model for platform, test experts, and tools
- Shared multi-tenant test environments
- Security (test user IDs/SSO)
- Integration of on/off premises systems
- Access to a variety of tools, ensuring more test coverage because of multiple tools testing
- Performance/volume test
- Defect isolation
- Documentation of "out of the box" requirements
- Release management

Cloud Computing Architecture

As shown in Figure 1.3, the applications hosted in a Cloud environment are available to end users on demand.

Key Drivers in Cloud Adoption

- SaaS (Software as a Service) model leading to TaaS
- SI (system integrators) play a key role in providing strategic and tactical support on their Cloud Initiatives
- Identifying components (infrastructural, platform, application, tools and business process level) that are suitable to be migrated to cloud

- POC – Proof of concept is a good initial approach in tactically connecting up the client's Cloud initiative
- The most important part of any Cloud offering is the service level agreement (SLA), which outlines your provider's guarantees. You will want to make sure that the SLA covers performance, uptime, notification of downtime, and other critical factors, as well as the repercussions for failing to meet those guarantees. The SLA should also specify the data model and rules for how data is to be de-normalized or represented.
- Choosing the right architecture – it is essential to use the right architecture for the SaaS application based on the criticality of your application and the appropriate level of security required. Ensure your data architecture supports multitenancy. In multitenancy, the same instance of software runs on the vendor's servers serving multiple clients (tenants).

Business Verticals

Cloud computing has become a common delivery model for many business applications and is being used widely in accounting, collaboration, customer relationship management (CRM), management information systems (MIS), enterprise resource planning (ERP), invoicing, human resource management (HRM), content management (CM), supply chain management, online backup, business intelligence (BI), service management and many more.

Approach to Cloud Testing

There is no well defined, ready-to-go approach for Cloud testing, since every business has its own set of requirements. However, the following established best practices will help to ensure success.

- Analyze if your business application is flexible enough to run in the Cloud.
- Understand the challenges you might encounter when moving to the Cloud.
- Selecting the right vendor is definitely important as it will decide robust your application is going to be and the financial viability, i.e. you can calculate the total cost of ownership (TCO), etc.
- Good understanding of the testing tools available and their pros and cons for testing different types of Cloud applications. A proof of concept will help in selecting the right automation tool,

Business Testing	Security Testing	Performance Testing	Compatibility Testing	Live Testing
Manual, Automated, Functional Testing Performing functional testing	**Application Security Testing** Cross-Site Scripting, SQL Injection, HTTP Header Injection	**Scalability Testing** Evaluate the behavior of system by increasing number of concurrent users to identify the breakeven point	**Multi-browser Compatibility** Validating if application looks and functions similar across all hardware and software configurations (Ex. browsers, OS, etc.)	**Disaster Recovery Testing** Check whether system is recovered successfully in an alternate environment due to an unexpected failure
Exploratory Testing Simultaneous learning application business, test design and test execution	**Network Security Testing** Firewall, Network-Based Anti-Virus, VPN Encryption, Controlled Access to sites and servers, etc.	**Volume Testing** Validating behavior of application when subjected to large volumes of data	**Localization Testing** Product designed should be customizable to be released across the globe	**Live Upgrade Testing** Ensure live upgrades do not impact existing connected SaaS users
Business Workflow Testing Thorough testing of end-to-end product	**Role-Based Access Control Testing** Test roles and privileges in a multi-tenant environment	**Availability Testing** Determining the stability and expected downtime of deployed application	**Accessibility Testing** Test to ensure if people with disabilities can access application effectively	
Automated Regression Testing Very critical in validating the frequent SaaS upgrades	**Data Integrity Testing** Testing data integrity among multiple tenants, their upstream and downstream systems and identify vulnerable scenarios, etc.	**Reliability Testing** Measuring performance degradation over longer periods at varying load levels	**Internationalization Testing** Product designed should function properly when customized for use in different languages and locales	
Data Migration Testing Validate different data sources from existing system that need to be migrated to SaaS application	**Compliance Testing** Testing CPI Compliance (PCI is a security standard for accepting credit cards), issues with cookies, SSL configuration, etc.		**Interface Compatibility Testing** Validate upgraded interface functions properly and also ensure if older interface of the SaaS application is good?	

Figure 1.4. Different Types of Testing in the Cloud

because automation plays a key role due to the agility of the Cloud applications. Apply the 20-80-20 rule, i.e. automate 20 percent of scripts that take 80 percent of execution time and 20 percent of the test engineer's skills.

- Prepare a roadmap, i.e. test plan on how to test applications in the Cloud

Classification of Different Types of Testing in the Cloud

The different types of testing shown in Figure 1.4 can be performed using the Cloud-based testing solutions.

- Functional testing –
  testing types specified under business testing
- Non-functional testing –
  testing types specified under security and performance testing
- Operational testing –
  testing types specified under compatibility and live testing

The Cloud Testing Workflow

Figure 1.5 clearly sets out the process to be followed to test applications in the Cloud.

Assess the functional, non-functional and operational test requirements
Understand the business logic thoroughly and prepare SRS

↓

Understand the usage patterns
Subscription can be monthly/quarterly/yearly

↓

Test strategy, plan and design
Identify the testing methodology, testing types, prepare manual/automation test plan and different testing design techniques to write manual/automation test cases with high test coverage and quality

↓

Prepare test cases and suite
Start writing the manual and automation test scripts for functional, non-functional and operational requirements if any

↓

Prepare test environment
QA/Staging/Pre-Prod which mimics the Prod environment

↓

Populate test data
Good thing with SaaS is we can test with real-time data, however appropriate security measurements to be taken care

↓

Execute SaaS testing, report and publish
Perform testing of application by executing manual and automation test scripts, and generate reports and publish to the management

↓

SaaS certification
Get your SaaS application certified by UptimeSafe, a service company which certifies the uptime and reliability of services offered by a SaaS SaaS certification service will test a cross section of SaaS provider's application network and certify their uptime based on network average

Figure 1.5. Testing in the Cloud – Workflow

Challenges

- Security, privacy, data integrity, and availability.
- Testing Cloud applications has its own set of nuances and challenges, such as SaaS upgrades, short QA validation cycles, testing live upgrades, data integrity, impact on multiple subscriber organizations, and, above all, the primary demand of high quality.
- Limited options for open source test tools.
- Testing Cloud applications and networks demands a wide mix of application traffic, current security coverage, and incredibly high performance and throughput.
- Thorough business knowledge for effective testing of configurable and non-configurable components.
- Validating interface compatibility, complying with government regulations and standards, such as PCI, etc.
- Simulating live upgrade testing.

Testing Tools

Cloud Testing Tools	Multi-Browser Cloud Testing Tools
CloudTest – SOASTA is the leading provider of cloud-based testing services and created the industry's first browser-based website testing product, CloudTest. CloudTest can be used when the stress testing, performance testing, and reliability testing needs to be performed on an SaaS application.	**Adobe Browser Lab** – A free online service that helps ensure your Web content display as intended across all browsers and operating systems.
PushToTest – A single integrated platform that enables unit tests to be reused as functional/integration/load/performance tests, and business service monitors. Supports automation of Web rich Internet applications (RIA) using Ajax techniques, SOAP/WSDL-based Web services, service-oriented architecture.	**Litmus** – In addition to cross browser testing, it also tests multiple e-mail clients such as outlook, gmail, etc. to ensure they display properly across all clients. Plug-ins allow for both Windows and Mac compatibility.
Gomez – The only Web, Cloud, mobile, and streaming application performance monitoring and load testing solution.	**Cross Browser Testing Tool** – In addition to cross browser testing, it also provides testing capabilities, such as checking their AJAX, JavaScript, and jQuery actions.
uTest, QMetry, PractiTest, LoadRunner, LoadStorm, Appistry, and many others.	

Advantages

- The main benefit with Cloud solutions is that they provide readily available business applications with an extremely short time to market.

- The low start-up costs, no maintenance, and rapid deployment of Cloud solutions such as SaaS have led to a rapid return on investment and SaaS is expected to be the highest revenue-generating service within the Cloud model for the next several years.
- Security is given top priority by following the most stringent quality standards to ensure that valuable information is protected against all types of threats.

Future of Software Testing – Cloud

- SaaS and other Cloud computing applications are expected to expand dramatically in the coming years for all small, medium, and large businesses.
- The rate of SaaS adoption is on the rise, as evidenced by the growth in worldwide SaaS revenue.
- Forrester, a global research and advisory firm, states that, going forward, more than 80% of US cloud revenue will come from SaaS.
- With global connectivity becoming the norm in technology, accessing applications via Cloud computing has a greater chance of gaining credibility because people will be accustomed to the on-demand exposure that SaaS provides.

Conclusion

- Cloud-based applications, such as SaaS, help organizations focus on their core business, rather than non-core areas such as IT application development, procurement of hardware/software, etc.
- They also reduce the effort required in supporting, maintaining, and upgrading these non-core applications, helping the organization to free up a significant amount of their management/ resource bandwidth.
- Automation of Cloud applications helps shorten the release cycle of frequent cloud application upgrades and releases.
- A comprehensive testing effort and a competent, skilled team is required to conduct the tests with the right strategy, so that organizations can leverage all the benefits of Cloud, such as higher system availability, greater reliability, higher flexibility, scalability, and enhanced levels of security.

Appendix

Acronyms and Abbreviations

- SAAS: Software as a Service
- TAAS: Testing as a Service
- ROI: Return on Investment
- MTTR: Mean Time to Market
- SLA: Service Level Agreement
- TCO: Total Cost of Ownership
- PCI: Standard for accepting credit cards
- QA: Quality Assurance
- WSDL: Web Services Description Language
- SRS: System Requirement Specification

Reference Material

- Protel white paper on SaaS: Software as a Service

> about the author

Varsha Jadhav – Currently a Senior Quality Analyst in the Tax & Accounting division of Thomson Reuters, she has around 6 years of experience in software testing. Prior to Thomson Reuters, Varsha worked with Verizon Data Services India Pvt. Ltd. for 3 years and Accenture Service India Pvt. Ltd. for almost 2 years. She has a master's degree in Computer Applications (MCA) from Osmania University, Hyderabad.

Testers, Developers, Consultants or Senior Consultants for Consultancy Services (m/f) for Agile software development and software testing

Grow with us – we are recruiting!

We are looking for dedicated and qualified colleagues to strengthen our IT Management & Quality Services team.

We are a market-oriented and innovative company based in Berlin with 40 employees. Apart from our high-value consultancy services in the areas of Financial Services, IT Management & Quality Services, Training Services und IT Law, our customers also appreciate the international conferences and publications on the subject of IT quality, which are organized by our Events & Media division.

Profit from our experience. We offer

- interesting and challenging IT projects in the area of IT processes and projects, staff qualification and coaching, software testing and test management, as well as architecture and security
- a direct working relationship with the division leader and the team leaders of IT Management & Quality Services
- career development within a flexible company

You're looking for something special? So are we!

Let yourself be infected by the friendly working atmosphere in a strong and motivated team.

You have

- a thorough education or have studied (e.g. information technology, business studies, mathematics) as well as several years of experience as an IT consultant in a business or organizational department of a company or as a consultant in appropriate projects;
- experience with Agile development models and techniques, as well as test procedures and tools used in this area;
- experience with change processes from traditional development to Agile methods;
- experience as a ScrumMaster, Product Owner or in a similar position;
- knowledge of the practical application of techniques and standards, such as CMMI®, SPICE, ITIL®, TPI®, TMMI®, IEEE, ISO 9126;
- sales experience and you recognize innovative sales methods.

You possess

- self-initiative and a good presence
- high willingness to travel
- good written and spoken English

Apply

Please send us your full and relevant application documents, including your salary requirements, only by e-mail to: *hr@diazhilterscheid.de*

Díaz & Hilterscheid Unternehmensberatung GmbH
Bereich Personal
Kurfürstendamm 179
10707 Berlin
Germany

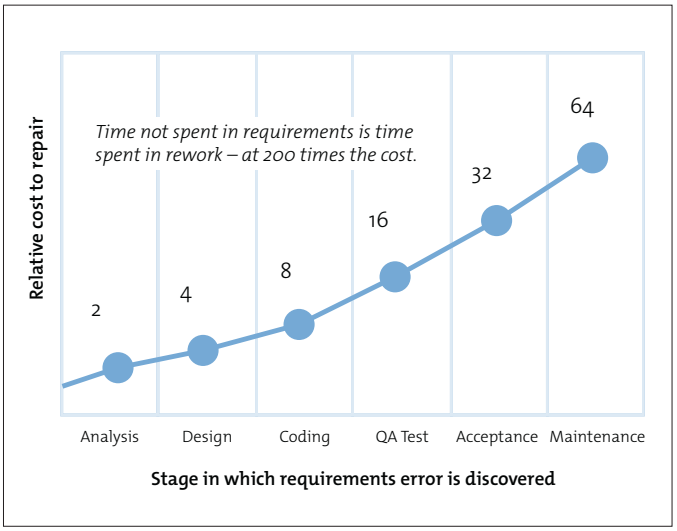
Phone: +49 (0)30 74 76 28-0
Fax: +49 (0)30 74 76 28-99
E-mail: info@diazhilterscheid.com
Website: www.diazhilterscheid.com

By Mahwish Khan & Rabia Akhtar

Indentifying and Fixing Requirement Defects in Later Stages Increases Costs

Abstract

The core focus of this paper is to inform and enhance the view of software process management and metrics, and dynamic process modeling. Our detailed analysis will mainly examine the impact of identifying requirements defects in the testing phase in terms of cost. There are defects in each phase of software development and the requirements engineering phase is a major source of defects. Requirements defects identified in the testing phase are the most costly to correct because they usually impact all phases of software develoment. Between 40 and 60 per cent of all defects found in software projects can be traced back to poor requirements [6]. According to Boehm, errors found late in the development process can cost up to 100 times more than if they had been detected during the requirement analysis phase [5]. Requirements defects occur due to incorrect or misunderstood requirements, or invisible requirements that result in reworking. Rework costs constitute between 70 and 80 per cent of development cost [2]. We will model these facts and prove that idenfying and fixing requirement defects in the testing phase increases costs.



Graph 1. *Relative cost to repair*

Index Terms – requirements defects, impact of requirements defects, testing costs, defects costs

1. Background

There is a big difference between requirements defects and other defects and these will be discussed in this paper. Defects in software products fall into two main categories:

- Implementation defects
- Requirements defects

Our purpose is to deal with requirements defects, the kind of consequences they have, and their impact on the costs of fixing any requirements defects at a later stage, such as design, code, or testing. Soren Lausen et al conducted a case study and they obtained 200 reports for this project. Of these, they analyzed 107 reports in detail and concluded that [2]:

- 65 concerned tacit requirements (requirements that had not been written down).
- 20 concerned requirements that were wrong, misunderstood or forgotten.
- 20 concerned external software packages or misunderstandings about how these worked. These were the most costly defects to repair.

Defects of both kinds may be detected at various stages of development. The earlier they are detected, the easier they are to repair. Ideally, they should be prevented from creeping in [3].

In this paper we only discuss requirement defects because they are more costly to repair than implementation defects. Due to their nature, we need to use other techniques to prevent or detect them before implementation. Programmers can, for instance, find implementation defects through testing or inspecting each other's programs, but they can rarely find requirement defects that way. Usually the environment must be involved in order to find these defects.

In the figure below it can be seen that late detection of defects results in rework that directly increases costs.

This figure is taken from the Boehm and Barry book "Software Engineering Economics" and the Boehm and Basili IEEE paper "Software Management".

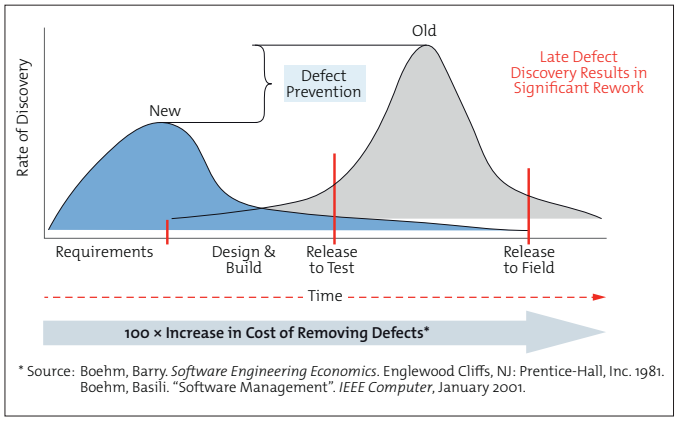


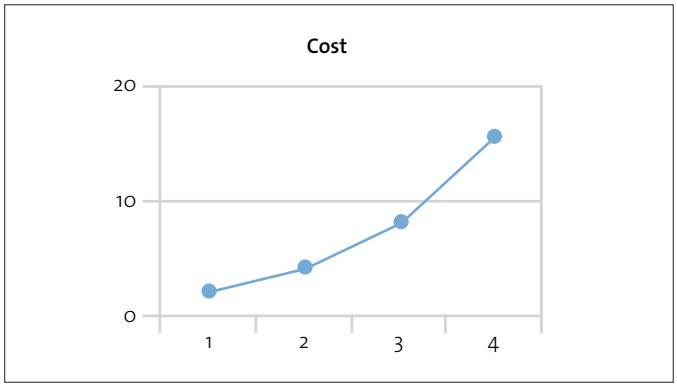
Figure 1. Rate of discovery of defects with time

2. Planned Software

Ideally, as we move ahead in the development life cycle, identified defects should decrease. Table 1 sets out the ideal cost of identifying and fixing defects in with regard to the SDLC phases. According to Boehm, if we identify a defect at later phases in the SDLC, its cost increases exponentially.

Actual cost of identifying and fixing defects			
Days	**Phase**	**Cost**	**Defect Detection Rate**
20	Requirement	2	40
40	Design	4	30
60	Code	8	20
80	Testing	16	10

Table 1. Nominal cost of Identifying and fixing defects



Graph 2. Nominal cost associated with defect detection rate

3. Actual Software

In the software industry, there are very few cases where practices have been following in the true sense. Requirements engineering is not done in the way it should be. That is the reason whythe defects were missed in each phase. The undetected defects included in the design phase result in design defects and subsequent code defects. All the defects are then left for identification at the testing stage, but this is very costly. If defects are identified at the end of the testing phase, it is necessary to repeat a complete cycle in order to correct the defects, and this results in rework.

Actual cost of identifying and fixing defects			
Days	**Phase**	**Cost**	**Defect Detection Rate**
20	Requirement	2	10
40	Design	4	20
60	Code	8	30
80	Testing	16	40

Table 2. Actual cost of identifying and fixing defects

4. Model Overview

The model covers the activities from design through to testing, including inspecting design and code, and is shown in Figure 2. The ITHINK modeling package is used to implement the system dynamics method [7]. In industry, system testing and inspections are the only defect removal techniques.

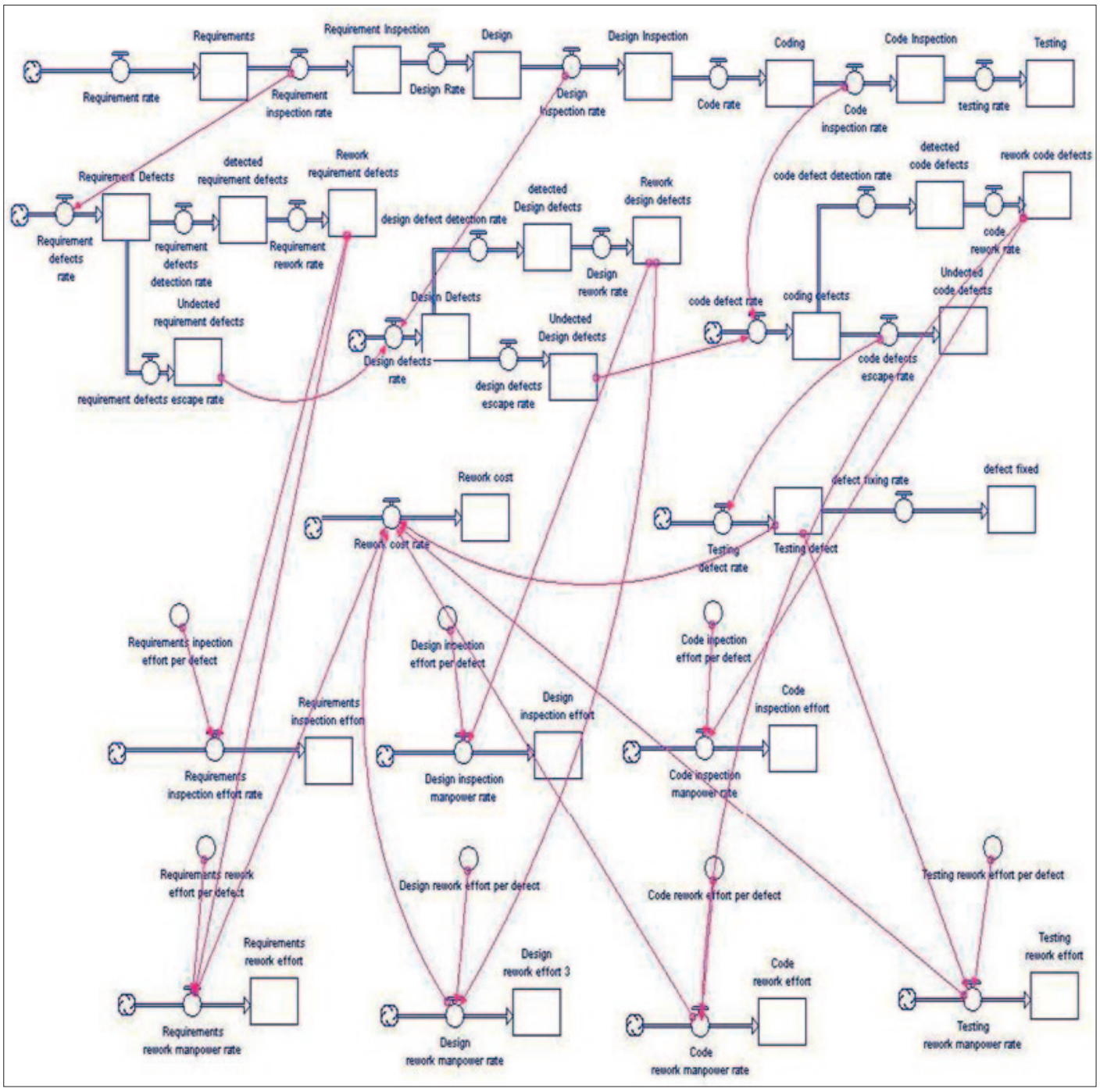


Figure 2. Software dynamic model

5. Model Description

Defects are generated through a co-flow of design and coding activities, and eventually detected by inspections or passed on to the next phase. Detecting defects is a function of inspection practices and inspection efficiency. All defects detected during inspections are then reworked. It should be noted that undetected requirement defects are amplified into design defects which are, once again, amplified into coding defects. Those defects that are still not detected during code inspections are then detected and corrected during integration and testing activities. The effort and schedule for the testing phase is adjusted for the remaining defect levels.

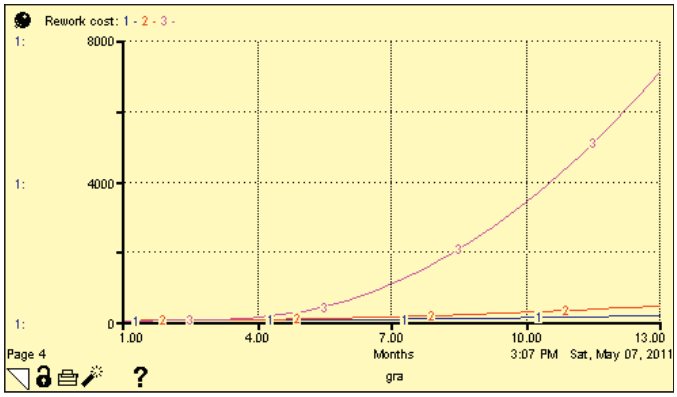
We also consider the inspection effort from requirement to testing and its impact on the number of identified defects. We then consider the rework cost for each defect, multiply it by the number of defects found in each inspection phase, and combine the rework effort across all phases to get a total rework cost.

6. Comparative Analysis

In our model, we have added inspection after every stage to detect defects. We have run the model for three different cases. These are:

1. All defects are identified at the requirement inspection stage (ideal case).
2. All defects are identified at the testing stage (worst case).
3. Different percentages of defects are identified at different stages, i.e. 40% of defects are identified at requirement inspection, 30% at design inspection, 20% at code inspection and the remaining 10% at testing (nominal case).

We have analyzed the rework cost for different cases and identified how the percentage of cost increases if all defects are identified at the testing stage. We have done a comparative analysis of rework costs in the above three cases.



Graph 3. Comparative analysis

In Graph 3, the first case is shown by rework cost 2. The second is shown by rework cost 3 and the third by rework cost 1.

As shown by our analysis, if all defects are identified at the requirement phase then the cost is $150, for 'rework cost 3' the cost is $ 6000 and for 'rework cost 1' the cost is almost $400. If we analyze this closely, we can say that the rework cost is increased by 4000 percent in the worst case compared to the ideal case and there is a 1500 percent cost increase in the worst case compared to the nominal case.

	Ideal Case	Nominal Case
Worst case	4000%	1500%

Table 3. Cost of rework for worst case vs ideal and nominal cases

7. Software Dynamic Model

See Figure 2, "Software dynamic model", 45.

8. Conclusion

The dynamic model successfully demonstrates several phenomena of software projects, including the effects of inspections. It has also been shown that the model is scalable to project size and calibratable for productivity, defect generation, and defect detection.

The model illustrates that, under ideal conditions, costs decrease because there is less rework and, in nominal conditions, costs are comparatively less than in the worst case as there is a little less rework, but in the worst case, costs are at their peak due to having the highest rework effort rate. However, the cost effectiveness of inspections depends on phase defects injection rates, defects amplification, testing defects, fixing effort, and inspection efficiency.

References

[1] Boehm, B., 1987. Industrial software metrics top-10 list. *IEEE Software*, 3, 9, 84–85.

[2] Lauesen, S. and Vinter, O., 2000. Preventing Requirement Defects. *Proceedings of the Sixth International Workshop on Requirements*, Stockholm, REFSQ'2000

[3] Lauesen, S. and Vinter, O., 2001. Preventing Requirement Defects: An Experiment in Process Improvement. *Requirements Engineering Journal* 6, 37-50. Springer Verlag.

[4] Jones and Capers, 2007. *Estimating Software Costs*. McGraw Hill, New York.

[5] Boehm, B., 1981. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, NJ.

[6] Davis, A. M., 1993. *Software Requirements: Objects, Functions, and States*. PTR Prentice Hall, Englewood Cliffs, NJ.

[7] Richmond B et al., 1990. *ITHINK User's Guide and Technical Documentation*. High Performance Systems Inc., Hanover, NH. ■

> about the authors

Mahwish Khan has 5+ years of high quality experience in QA engagements, process, methodologies, test automation, performance testing and business UATs, as well as focusing on web and mobile applications. She deals with defining test system architecture, converging systems after acquisitions, managing people, implementing process improvement and change management, project management, data driven decision making, and setting up best practices. Starting as a test engineer, she soon started to coordinate and manage test projects at different companies and organizations. Parallel to that, she has expanded her knowledge of usability testing and accessibility.

Rabia Akhtar is Senior Software Engineer at Virtual Soft, Lahore, Pakistan. She has more than 5 years of experience in complex software development. She has a technical background involving various different programming languages and software quality assurance tools. She has acquired a depth of knowledge, especially in the areas of web development, quality management of web applications, fault management, and automated unit testing.

Training with a View

IREB International Requirements Engineering Board

CAT Certified Agile Tester®

ISTQB® Accredited Training Provider

More dates and on-site training worldwide in German, English, Spanish and French on our website:
training.diazhilterscheid.com

Kurfürstendamm, Berlin © Katrin Schulz

By Marc van 't Veer

What is testing in production and why should it be performed?

Based on the experience from a project without TiP, reasons are given why the DTAP model is not suitable for testing modern public online services in the cloud. Apart from TiP being necessary, this paper also argues that TiP should be a regular test level. By comparing a test and the production environment, a view is given on how testing in production should be perceived as normal. Additional to this, a definition of TiP is given and the paper ends with questions for future research.

Introduction

Would you test differently if you could Test in Production (TiP)? Would you follow a different strategy? In a project a public web service was created without TiP. Based on the experiences gained from this project, reasons are given why a test and acceptance environment from the DTAP model (Development, Test, Acceptance, and Production environment) are no longer sufficient (Aalst, 2006), (Pol, 2000). More generally, traditional test environments are not suitable for testing modern public online applications, such as web services in the cloud (Grundy, 2012). Besides that, TiP is a necessity and it is also argued that TiP should be a regular testing level. Based on the comparison between test and production environments, a view is given on how normal it should be to test in production. Additionally, a definition of TiP is proposed and the paper ends with questions for future research.

The natural environment of the new public web service is illustrated using a test project performed at a large Dutch mobile telecoms provider. This service is an API and will be used in the future by external apps and on social platforms such as Facebook (API). An application (app) is typically a small, specialized program downloaded onto mobile devices. With the publication of the API in the cloud, multiple platforms can be supported without creating apps for them all. Using an app, and the web service behind it, end users can access their call and internet credits, address data, and invoices on their handsets in real time. The project needs to ensure the API is stable, has a high performance, is strictly secure, has a re-usable infrastructure and is easily extendable to cope with new functionality. All these requirements determine how the API will perform as a public web service for external users. Users are not only apps but also external developers. A few examples of the risks test management perceives are: unknown integrations, unknown diversity in contract data, and how this data will be presented. The API has no influence on the presentation to the end user. It is possible to use the API wrongly, whether deliberately or unintentionally. Can a tester be held responsible for defects between the API and an external app?

An end user has no clue about the number of steps between the app on his handset, the API, and other back-end systems. For him or her, everything behind the app is the cloud (forum). The end user will judge the quality of the API by the quality of the app. Without an early integration between the app and the API, all possible integration defects will become visible when the API is in production. The test strategy for the API project was "early integration test with complete infrastructure". This strategy was implemented by dogfooding during development, and by system and integration testing. The API was developed using prototypes of the app. Examples of the prototype apps are SoapUI, a web browser, a Windows taskbar app, and a development app which all use the API (dogfooding). During acceptance level, a first prototype of the external app was tested within the internal network. The integration tests were completed with a test of the process to download, install, and log into the app on a private mobile production network.

The moment the app was live and the API was used, the first incidents were reported. For example, a customer had a rate plan that contained two bundle types: calling and text messages. But for each rate plan, only one usage progress bar (only one bundle type) could be shown in the app. A second example was that it was unclear to many end users how to log into the app. Customers were using their existing login account instead of a special access code. A third example was, if the customer had too many wrong login attempts then his account would be temporarily blocked. Here the API was using the standard HTTP error codes like "401 – Unauthorized" (error codes), which is returned if a customer cannot log in. But the app wanted to show a different error message for the cases where the user was inputting a typo and when there were too many wrong attempts. Unfortunately, this difference could not be determined from error code 401. Now the standard error code is extended with specific API error codes: 401 – Unauthorized + 1102 – Invalid username or token of 401 – 1103 – Account has been temporarily locked out.

The app and the API were being fully integrated for the first time in the production environment. Just before going to production, the idea was proposed to perform a number of tests in production because only then would a representative *test* environment become available. People were surprised by the idea and were against it because *"testing in production is simply not done"* (TestNet, 2012), (Groot, 2012). Within the project and with the mobile telecoms provider it was not possible to test in production. In the end, all tests need to provide insight into the risks and find a way to reduce them. By adding TiP as a test level, the behavior and the possible risks of running in production for this service could be made visible.

In the introduction a test and production environment are mentioned, but what exactly are they and what are the differences?

Environments

A test environment is described as "an infrastructure that is needed to perform a test" in (Pol, 2000) and (Aalst, 2006). A test environment should be stable, controlled, and representative. There can be multiple types of environments with different roles. TMap and TMap-Next define a test environment as "a combination of hard and software, connections, environment data, administration tools and processes wherein tests are being executed" (Pol, 2000) and (Aalst, 2006).

During the development cycles of an application, every environment within the DTAP model is used by different consumers. White-box tests are executed on the development environment, the first application tests are performed on the test environment, and the acceptance tests are executed on the "as-it-were-production" environment. Because the usage is distributed over these different environments, development can continue without interfering with an acceptance test (environment). The maintenance organization has the responsibility to keep existing applications up and running on the production environment (Aalst, 2006). With each next level of the DTAP model, more systems are connected to each other and, thus, the environment becomes more comparable to the production environment. As a result, the applications become more robust and mature. The production environment is, thus, protected from low quality and remains stable.

When you look at the context of where the API and app need to operate, as stated in the introduction, a test environment is representative if it is **not** stable. Changes are continuously introduced, connections may drop, users are all external, the devices are very diverse, and there are connections being used that did not exist when the API was developed. The API's natural environment is a publicly controlled domain. In Blokland, 2012 and in Whittaker, 2011 an overview is given of how dynamic, complex, and diverse the environment of a modern public service is. If the goal is that a test should be executed in a representative environment, then this can only be done in production (Blokland, 2012).

A production environment is often compared to a test environment (production-environment). Production is not just an environment but the location where an application should perform "for real". Comparing a test and production environment to each another is like comparing a zoo to nature ("The wild" as referred to in the animated movie Madagascar (Madagascar)). They may have similarities but the differences are plentiful. Also, the notion of *"environment"* sounds too much like a clear ordering of the context where the application is going to be used. In production the service is running "live" and the API and app are servicing the end user. The test environment is a parallel world where changes and new projects are being developed without any impact on the live service. The environment where the service is running is critical for the overall quality. If the test and production environment are more similar, the test results say more about the quality and possible risks. Microsoft's Seth Eliot made a very clear statement when he discussed the "Google Staged Approach" (Eliot, 2012, spring STP). This approach follows four test levels and three of them are in production. The first level is the traditional "upfront tests" in a test environment; the other three levels are user scenario execution, dogfooding, and beta testing in production. Google is going fast to production because there is no added value in more upfront tests. Ten percent more upfront tests does not result in a ten per cent improvement in software quality (Whittaker, 2012), (Eliot, 2011-spring QASIG).

For some it may be "*not* done" to test in production, but what is the value of the test results if the traditional test environment is not representative? Should TiP be a normal test level?

How normal is testing in production?

If software is compared to a "normal" product or service, then it is clear that testing in production is not new. For normal products there are always tests in production. The Dutch Organization for Applied Scientific Research performs production tests for the introduction of the 4G LTE technology for fast mobile Internet in the Netherlands (TNO), (TNO-Huawei). The consumers' union also performs production tests on all kinds of products and services (consumentenbond). When making a little side step into law and liability, then there are more arguments to find why it is normal to test in production. Software is more and more a daily good of the end customer and they expect the software to be reliable. The moment an incident is found, the supplier is responsible. Levy and Bell did research on product liability and what actions can be taken to minimize, transfer, or divide the risks (Bell, 1990). With a legal action against a supplier there is the possibility to make the rights and obligations between the supplier and the customer clearer. The insight into possible risks and liability contributes to *"a positive relation between the supplier and his customers"* (Bell, 1990). By creating jurisdiction about rights and obligations, expectations are more realistic and there is a greater chance that possible problems in the software are discovered, admitted, and corrected before any loss or damage is done (Bell, 1990). The law is different on products and services. The sale of products is subject to many damage and warranty laws (Bell, 1990). These laws do not apply to the sale of a service. The website of ICTRecht (ICT and law) gives basic information for software developers (ICTRecht). This information also includes how to handle liability cases. A developer can be liable for incidents that occur in production (ICTRecht). Software will become a "normal" product and customers expect that the same laws on damage and warranty will become applicable. When looking at product quality, a product should work in production as specified.

Continued product quality control does not stop the moment a product is in production. Within the traditional V-model for software development, each next phase is validated by a test level and so the quality is monitored (Aalst, 2006). The last test level in the V-model is a production acceptance test (PAT), which is generally done by the maintenance organization. In the model there is no test level defined after the software is accepted. ISTQB added a "Maintenance and operation" phase to perform maintenance tests. Maintenance tests are also called "in-service inspection and testing'" (maintenance-testing). These tests focus on regression defects and are comparable to a yearly car checkup (General Periodic Inspection or, in Dutch, "APK"). If a car does not pass the check-up then it needs to be repaired if it is to continue on the road. If you look back at the V-model, these "in-service" tests should be a part of the model. When these tests are added to the model it becomes a circle, since the expected and the real behaviors are compared in production. If you look at the developers at Google, you see that they have much more direct contact with the end users than traditional developers, since Google follows a fundamentally different development model ("Google Staged Approach") for cloud services. Traditionally, the requirements of the end users are constantly translated between the developer and the tester because there is no direct contact between them. According to Whittaker, 2011 there will be

much more testing in production in the future and the whole development model will change.

TiP is necessary to perform good tests for modern public online services such as an API. But what is TiP exactly, and what is its definition?

Definition of testing in production

To have a better understanding of what Testing in Production is, a brief idea is first given of what TiP is and what it is not. TiP is not a replacement for any test level such as system or acceptance tests. This means that TiP should not be done because there was no more time left or because it's cheaper. The goal of TiP is not to check whether an installation was correctly performed on production. TiP is not a production acceptance test, because these tests are done *before* the software goes to production and their goal is also different. A PAT looks at whether the software is easy to maintain rather than how it is used or behaving in production (Aalst, 2006).

TiP is described as testing of software which is installed and executed on the hardware of the end user and is used for real. Wiki defines TiP as follows:

'Production testing is when you are testing a real live system, either about to go live or with live users. It is needed because having software working on the developer's computer is no guarantee that it will work in the client's installation.' (TiP)

Seth Eliot defines TiP as follows:

'Testing in production (TiP) is a set of software testing methodologies that utilizes real users and production environments in a way that both leverages the diversity of production, while mitigating risks to end users.' (Eliot, 2012)

Important elements from these definitions are that tests are done by or for *end users* on *real live systems* with the focus on its *behavior* in its diverse environment.

With the implementation of TiP testing methodologies and techniques that use the diversity of the production environment, the risks of end users can be minimized. The diversity of the production environment functions as a lever to find defects that can only be found there. When going back to the central question, *Why should we test in production?* We can see that there is not only something missing in the definitions (for which types of application is TiP applicable) but they focus too much on methodologies. Eliot is explaining that TiP is meant for (web) services (Eliot, 2011) and defining TiP as a test *type*, while the reasoning in this paper is that TiP is a test *level*. A test level is a group of test activities that are combined, executed, and managed (Aalst, 2006). A test type is a group of test activities that tests the software or a part of it on a combination of quality attributes (Aalst, 2006).

Testing in production can be compared to system testing. A system test is done by a supplier on a lab or test environment, with the goal that the developed system complies with the functional and non-functional specifications, and with the technical design (Aalst, 2006). The difference is that TiP is using the production environment and that behavior is inspected with the possible edge cases that occur naturally there. These edge cases are unthought-of and unpredictable. By using the production environment with real users, this diversity is used for testing. With system testing the focus is on expected behavior and with TiP the focus is on unexpected behavior. Within this new test level, all the test methodologies of Eliot can and should be used like synthetic tests in production or controlled test flight (Eliot, 2012).

When combining all previous points then the total definition of TiP becomes:

'Testing in production is a combined group of test activities that uses the diversity of the production environment and real end user data to test the behavior of the developed service in the live environment, so minimizing the risks for the end users.'

When TiP is introduced into an organization, the whole development model changes. Before TiP is added then it is mainly BUFT or "big up-front testing" that is executed. This means that all tests are done before the software goes live (BUFT). But, with TiP, the developers have more direct contact with the end users and the testers do not need to act like end users. The test environment becomes an internal production environment where testing is done by dogfooding and bringing your own device. When the software is stable, then testing is done in production via beta testing, for example (Whittaker, 2012).

Further research

One of the first questions to answer is: if tests are executed in production, how can this be done without influencing the continuity and stability for the real end users? Other questions are: how can regression testing or test harness be used for monitoring and what does it look like? (Riungu-Kalliosaari, 2012) When a test strategy is put forward for TiP, there should be a discussion with the operations department because the monitoring of all systems can be influenced by TiP. Operations is constantly monitoring the production environment, but when testers are testing in production they are also monitoring both with their own goal. Between test and operations a new role can be created like *"Testops"* (Eliot, 2012). The moment TiP is executed, it must be clear which strategy can be used to cover the risks. If there is a difference in responsibility between fixing defects in the test environments and incidents in the production environment, what will happen when you find a defect in production? Who is going to fix it? Who is paying for this? And when should it be fixed? There should be more research done on how TiP can be introduced into an organization and which techniques can be used to find different types of defect.

End notes

Books

- Aalst, L. van der, Baarda, R., Broekman, B., Koomen, T., Vroon, M., 2006. *TMap® Next, voor resultaatgericht testen*,Tutein Nolthenius. p. 44-51, p. 68, p. 82, p. 412, p. 419
- Bell, S. Y., Levy, L. B., 1990. Software product liability: Understanding and minimizing the risks, *Berkely Technology Law Journal*, Boalt Hall School of Law, University of California at Berkeley, California. Vol. 5, issue 1, 2–3. Available at: *http://www.law.berkeley.edu/journals/btlj/articles/vol5/Levy.pdf*
- Blokland, K., Mengerink, J., 2012. *Cloutest, Testen van cloudservices*. Tutein Nolthenius. p. 162–170

- Eliot, S., 2012. *Testing in Production A to Z, TiP Methodologies, Techniques, and Examples, Software Test Professionals*, p. 5, p. 55, p. 19, p. 20. Available at: *http://www.setheliot.com/blog/a-to-z-testing-in-production-tip-methodologies-techniques-and-examples-at-stpcon-2012/*, *http://www.softwaretestpro.com/item/5477*
- Eliot, S., 2011. *Testing in Production, Your Key to Engaging Customers*, Quality Assurance Special Interest Group, p. 19–20. Available at: *http://www.qasig.org/presentations/QASIG_Testing_in_Production-Engaging_Customers.pdf*
- Groot, D. J. de, 2012. TiP and the iceberg, *Professional Tester*, June. Available at: *professionaltester.com*
- Grundy, J., Kaefer, G., Keong, J., Liu, A., Guest Editors' Introduction: Software Engineering for the Cloud. *IEEE Software*, March–April 2012, vol. 29, no. 2, p. 26-29, Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
- Riungu-Kalliosaari, L., Taipale, O., Smolander, K., Testing in the Cloud: Exploring the Practice. *IEEE Software*, March–April 2012, p. 46–51
- Pol, M., Teunissen, R., Veenendaal, E. van, 2000. *Testen volgens TMap®*, Tutein Nolthenius, 5th edition, p. 495
- Groot, D. J. de, Kramer, A., Loenhoud, H. van, Prins, A., Ro, J., Schoots, H., Vorst, P., 2012. *Bepaal je koers!, Toekomst en trends in testen*, TestNet, p. 75-81

Internet sources

(API)

- API definition: *http://en.wikipedia.org/wiki/Application_programming_interface#Use_of_APIs_to_share_content*
- Project API: *https://capi.t-mobile.nl/*
- Customer API documentation: *https://capi.t-mobile.nl/documentation*
- My T-Mobile app: *http://www.t-mobile.nl/service-en-contact/apps/my-t-mobile-app*

(app)

- Definition of app: *http://dictionary.reference.com/browse/app*

(BUFT)

- *http://blogs.msdn.com/b/seliot/archive/2010/01/20/building-services-the-death-of-big-up-front-testing-buft.aspx*

(dogfooding)

- *http://harveynick.com/blog/2011/08/26/dogfood-nom-nom-nom/* and *http://en.wikipedia.org/wiki/Eating_your_own_dog_food*

(error codes)

- *http://en.wikipedia.org/wiki/List_of_HTTP_status_codes*

(environment)

- Why are there so many environments? *http://askville.amazon.com/environments-software-development/AnswerViewer.do?requestId=2956413*

(forum)

- Comment on the T-Mobile Forum (in Dutch): *https://forum.t-mobile.nl/belstatus-398/de-online-belstatus-gadget-p-edge-werkt-niet-meer-151973/*

(consumentenbond)

- *http://www.consumentenbond.nl/over/Onderzoek/onderzoek_en_verslagen*

(ICTRecht)

- *http://ictrecht.nl/diensten/juridische-producten/starterspakket/softwareontwikkelaars/* and *http://blog.iusmentis.com/2011/09/19/ben-ik-aansprakelijk-voor-de-fouten-in-mijn-software/*

(Madagascar)

- *http://en.wikipedia.org/wiki/Madagascar_(2005_film)*, *http://www.imdb.com/title/tt0351283/*

(maintenance testing)

- *http://en.wikipedia.org/wiki/Maintenance_testing*

(production environment)

- *http://www.head-fi.org/t/167356/what-does-the-word-production-environment-mean* and *http:/www.techopedia.com/definition/8989/production-environment*

(TiP)

- *http://wiki.answers.com/Q/What_is_production_testing_in_Software_development#ixzz266VbdjZh*

(TNO)

- *http://www.tno.nl/groep.cfm?context=thema&content=markt_producten&laag1=897&laag2=191&item_id=377*

(TNO-Huawei)

- *http://www.nu.nl/internet/2601169/tno-en-huawei-testen-lte-in-nederland.html*

(Whittaker, 2012)

- *http://www.computer.org/cms/Computer.org/dl/mags/so/2012/02/extras/mso20120200004s.mp3*

(Whittaker, 2011)

- *http://blog.utest.com/testing-the-limits-with-googles-james-whittaker-part-i/2011/05/* and *http://blog.utest.com/james-whittaker-the-future-of-software-testing/2008/11/*

> about the author

Marc van 't Veer is a test consultant at Polteq and has well over 5 years of experience as a coordinator and system tester. He has gained a lot of testing experience in technically oriented contexts, such as telecoms, SOA, test automation, development of stubs and drivers, and testing API's. In his current project he provides a lot of training and he manages the outsourcing of testing to India. He holds a MSc. from the University of Eindhoven in the Netherlands and holds the ISTQB advanced certificate in software testing.

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.com
Website: www.diazhilterscheid.com

Díaz & Hilterscheid is a member of "Verband der Zeitschriftenverleger Berlin-Brandenburg e.V.".

Editorial

José Díaz

Layout & Design

Lucas Jahn
Konstanze Ackermann

Website

www.testingexperience.com

Articles & Authors

editorial@testingexperience.com

Advertisements

sales@testingexperience.com

Subscribe

www.testingexperience.com/subscribe.php

Price

online version:	free of charge	www.testingexperience.com
print version:	8,00 € (plus shipping)	www.testingexperience-shop.com

ISSN 1866-5705

Picture Credits

© iStockphoto.com/artvea	1	© iStockphoto.com/madisonwi	22	© Ljupco Smokovski – Fotolia.com	43
© Jacek Chabraszewski – Fotolia.com	7	© iStockphoto.com/konstantynov	24		
© iStockphoto.com/CathyKeifer	19	© iStockphoto.com/rappensuncle	27		

Index of Advertisers

SWE Guild

The Software Engineering Guild

www.sweguild.com

Certified Agile Tester

Pragmatic, Soft Skills Focused, Industry Supported

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

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

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

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

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

Book your training with Díaz & Hilterscheid!

Open seminars:

28.01.13–01.02.13, München/Stuttgart, Germany
11.02.13–15.02.13, Berlin, Germany
22.04.13–26.04.13, Frankfurt/Karlsruhe, Germany
13.05.13–17.05.13, Köln/Düsseldorf, Germany

(German tutor and German exam)

14.01.13–18.01.13, Amsterdam, The Netherlands
11.03.13–15.03.13, Helsinki, Finland
08.04.13–12.04.13, Stockholm, Sweden
15.04.13–19.04.13, Oslo, Norway

Díaz Hilterscheid

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