12

December 2010

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Open Source Tools

ONLINE TRAINING

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

ISTQB Accredited Training Provider

ISTQB International Software Testing Qualifications Board

Díaz Hilterscheid

te testing experience

Editorial

Dear readers,

Open Source inherit a risk and this risk gets different weights by the implicated actors in the project or company.

I remember my first project working at the police IT-department. We found a very cool application developed by a very smart and small company in East Germany at the beginning of the '90.

They developed the software and hardware to analyze traffic control films. I know, we hate this pictures!! They used their own graphic controller and film reader hardware. For that they used Linux!

Today we would say, that Linux has not really a risk. There is a common open source OS supported by many companies and individuals. At the beginning of the '90 it was not really the case at all.

The IT directors of the ministry liked the solution but didn't want to have Linux. Too risky! The solution was that one of the biggest software and hardware vendors worldwide delivered the Linux PC in a rack of this company – you could see the logo! – and they were the company who sold this to the police under its flag. The small and smart company was subcontractors. The risk was minimized in the eyes of the IT directors. Today we can laugh about it, almost 20 years later, but it was quite stressful. We had a lot of fears...

Linux is and was not a small solution that is used by a "few" people and don't have the support of the community. There are other tools that are far away from this position. I think that one of the things to avoid and this is also valid for commercial software, is to minimize the risk. Risk mitigation paired with finance investment and expected ROI are probably the key to use Open Source software. Open Source tools are like life, they are born and they die, but you got the code! We do have some articles talking about all this. Thanks to the authors for taking part in this.

We are facing the next year, not only were we faced with Xmas decorations in October already, but we are also facing the planning of our events for the new year. We got over 90 papers for the Belgium Testing Days. Wow! Have a look at the program. We have a very good feeling about the success of the BTDs. The Call for Papers for the Testing & Finance Europe will run until December 31st . The leitmotiv of it is: The future of finance: Agility and security. Please send us your paper. As soon as we have dates for the Testing & Finance USA we will inform you. Don't miss the planned tutorials of Gojko Adzic and Michael Bolton in January. Have a look on http://www.testingexperience.com/knowledge_transfer.php

Sometimes you have friends that meet 4-6 times a year and spent time with them in the bar and talk about many things, not only about business and football. But sometimes you get an email and see that you really need more time to get to know them better. I got an email from my friend Geoff Thompson informing me that he is supporting Movember – a program that raises money for the awareness, public education, advocacy, support of those affected, and research into the prevention, detection, treatment and cure of prostate cancer. I'm really aware of this problem due to the death of my uncle, but also because another friend of mine has fighting this disease for the last two years. I kindly ask you to support this initiative. Part of the game is to laugh at my dear friend Geoff and how is face changed in the last 4 weeks. It reminds me of the days when I used to "wear" a moustache! Please have a look on the website http://uk.movember.com/mospace/554086/.

I get myself checked every year! Don't miss the date with your urologist! It could save your life!

Last but not least I thank the sponsors of the magazine, the authors and you the reader for spreading the word.

I wish you all a nice seasons holidays and enjoy the time with the people you love.

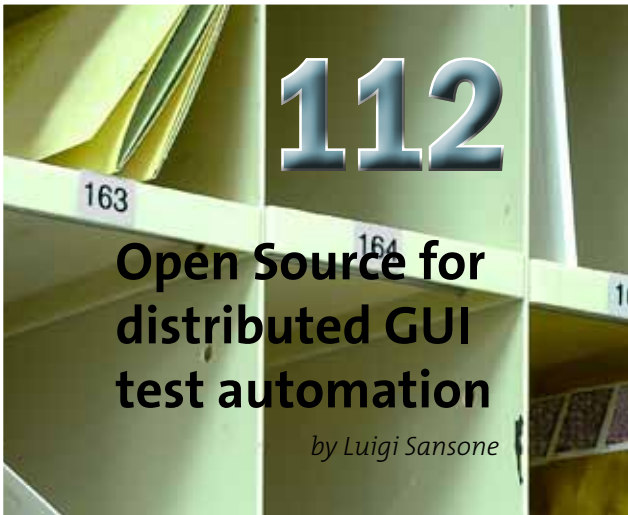
Enjoy reading

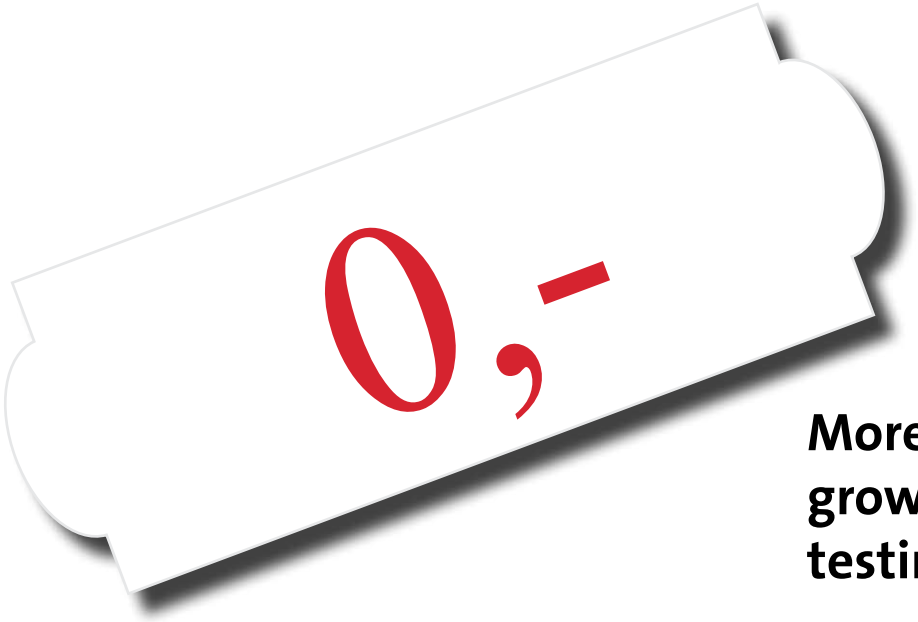
Contents

6

by Mark Aberdour

More for less: the growth of open source testing tools





More for less: the growth of open source testing tools

by Mark Aberdour

In the article we look at the how open-source software has grown from the domain of technology geeks to becoming an integral part of many organizations' software strategies today. As open-source software moves from infrastructure into the applications market, we look at what effect this will have on the major software testing tool vendors, followed by an overview of some of the most popular open-source testing tools and how to go about evaluating them.

Growth of open source

The recent recession has tightened the budgets of organizations the world over and most software and services companies have really felt the pinch. However, open-source companies have bucked this trend by exhibiting strong growth throughout. In the midst of the recession, an Economist article caught the zeitgeist with the headline "Open-source software firms are flourishing". In general there's been a real increase in open-source success stories in the media, while industry analysts like IDC and Gartner have been investigating open-source adoption and publishing results that show strong growth in the sector.

As long ago as 2008, Gartner surveyed 274 enterprise companies globally and found that 85% had already adopted open source, with most of the remaining expecting to do so within the following year. That prediction came to fruition. Forrester's 2009 report "Open-Source Software Goes Mainstream" questioned 2000 software decision makers and concluded that open source had risen to the top of the executive agenda, with the author noting that "the mandate's coming from higher up" and that many bosses are now demanding "faster, cheaper, better". This was a major step- change for open source, which had historically entered organizations under the executive radar from the bottom-up, installed by IT and programming teams and gradually gaining traction through the organization. That situation has now turned on it's head with the demand coming from the top. In a May 2009 report by Wall Street & Technology, Accenture echoed this, saying that „financial services firms are taking a different look at open-source technology now that they are financially constrained". This article cited major investment banks not just using but actually contributing to open-source projects, recognizing a further step-change and a maturity to open-source adoption that goes beyond simplistic cost savings.

Effect on commercial testing tools market

Much of the traction gained by open-source has traditionally been at IT infrastucture level. However, the other big change in open-source adoption in recent years has been in the move 'up the stack' from infrastructure into applications. Certain applications markets have already been significantly disrupted by open source: take SugarCRM in the CRM market and Alfresco in the document management market. The testing tools market is an interesting case, as open-source has clearly made significant progress. Opensourcetesting.org launched in 2003 and at the time had about 50 tools listed; nearly ten years down the line number has increased to over 450 tools. This increase in activity has been phenomenal and is far more than occurred in the CRM market, for example, before Sugar became the dominant application. However, the testing tools market is much more fragmented into many varied and specialist areas, which accounts for the large number of open-source tools. In a specific area, such as defect tracking there are only a handful of tools really vying for the top position.

The open-source testing tools market is also young and has not seen significant commercial investment yet. But regardless of that, expect the steady march to continue. The disruptive nature of open source in software markets does force commercial vendors to take stock of their position and the response must either be to innovate their product more rapidly to keep one step ahead, or to innovate their business model.

Microsoft Sharepoint is a great example of a commercial product that took the latter approach, in order to achieve the former. The core Sharepoint product is as closed-source as ever, however there are over 3,000 APIs available to programmers to enable them to extend the product in new and unforeseen ways by creating services, add-ons and integrated solutions. As noted on CMS Wire, this has resulted in a vibrant ecosystem and community to rival that of any open-source product, a shrewd move on the part of Microsoft. The success of Sharepoint will no doubt become more widely replicated in the years ahead by commercial vendors faced with open-source competition. It is quite possible that the likes of Mercury will move in this direction, which could make for interesting and innovative times ahead.

Quick overview of popular tools

On opensourcetesting.org the tools are broadly split into categories such as functional testing, performance testing, test management, defect tracking and so on. There are clear leading contenders in each category, but a whole host of other that are well worth your time investigating. Here I have cherry-picked some of the best-of-breed open-source testing tools currently listed.

Selenium - http://seleniumhq.org/

Selenium has forged a great name for itself in the functional testing market. A basic capture-replay tool gets you off the ground and will playback in many different browsers. You can then use the IDE to port your test scripts into a wide variety of programming languages to tweak them further, and can manage your script libraries through a number of different testing frameworks.

Cucumber - http://cukes.info/

Cucumber allows you to write test scripts in plain, human-readable text, not a programming language. The test scripts describe how software should behave in business language, so can serve as documentation, automated tests and development aid, all rolled into one. Cucumber works with Ruby, Java, .NET, Flex or web applications written in any language and has been translated into over 30 spoken languages.

Watir - http://watir.com/

Watir allows you to write easy to read tests with the emphasis on simplicity, flexibility and maintenance. It will do your basic capture-replay and will also check the expected results on a page. It supports web apps developed in any language and a wide range of browsers. A large community has developed around the product, which has become very well regarded over recent years.

Apache Jmeter - http://jakarta.apache.org/jmeter/

Possibly the oldest and most well regarded of the performance testing tools, Jmeter is a functionally rich tool for performance and load testing that has attracted a significant following over the years. The interface won't be to everyone's liking, but it's a very powerful and flexible tool that is well loved by many.

Radview WebLOAD - http://www.webload.org/

As a commercial testing tool vendor, Radview must be commended for taking up the open-source gauntlet, although the strategy has wavered over the years from initially going 100% open source, and later moving to the current 'open core' model whereby WebLOAD open source forms the engine of WebLOAD Professional, the latter offering many additional features and commercial support. There's no doubt it's a great product, but it hasn't really formed a developer community around the open-source tool to date, which other open-core business tools like SugarCRM have done successfully. It would be good to see WebLOAD move in this direction.

Bugzilla - http://www.bugzilla.org/

I tend to measure all bug trackers against the commercial tool JIRA which knocks the socks off anything I've ever seen. The open-source tools aren't there yet, but it's not for want of trying. Bugzilla has been going for what seems like decades and every software developer seems to have heard of it. It's neither the most graceful looking of the open-source bug trackers nor the easiest to set up, but it's probably the best in terms of functionality, flexibility and add-ins.

Mantis - http://www.mantisbt.org/

The other main contender in the bug tracking area and with a decent interface. It needs some technical ability to really configure it well, but that said, it's a good, reliable and well-proven tool out of the box.

TestLink - http://www.teamst.org/

TestLink does a very good job of providing a decent test case management tool. You can use it to manage your requirements, test specifications, test plans and test case suites. It provides a flexible and robust framework for managing your testing process, with a strong community and very active development team.

How to evaluate open-source tools

One of the questions often asked by potential open-source users is around the reliability of open-source software. Publicly available code for popular and mature open-source products tends to be reviewed by hundreds of developers, far more than most commercial software companies could afford to test their own products; so quality and reliability tend to be high.

Open-source development generally avoids many of the procedures that we normally take as software development best practices, without software quality suffering. For example, in open-source projects you will be unlikely to find detailed planning and scheduling efforts, deadline-driven milestones and documented risk assessments. What you will find is a rapid, iterative release process that results in continual improvement by large numbers of developers contributing iterations, enhancements and corrections. An academic study of 100 open-source applications found that structural code quality was higher than expected and comparable with commercially developed software, and the more well-known projects like Apache and the Linux kernel actually showed a substantially lower defect density than in comparable commercial products. So there's no question about it: open source can deliver top quality products.

However, how can you separate the good from the bad? A number of key areas result in high quality open-source software which are fairly easy to substantiate with a bit of web research:

- A large and sustainable community will develop code rapidly and debug code effectively, all open-source projects have community areas on their website where you can see numbers of registered users and gauge the amount of activity on forums, wikis and release pages, and readable source code repositories allow you to see how often code is committed.

- Modular code supported by good documentation will serve to attract new core developers and extension/plug-in builders, while documentation is visible on the project website. Just ask a developer to take a look at the code to assess its quality.

- A well-led core team that responds quickly to peer reviewers' comments and code contributions will result in rapid innovation and high quality being achieved. This will be visible through project forums and public mailing lists.

- Re-use of established and proven code libraries rather than writing the whole program from scratch helps drive high quality. These libraries should be obvious from reviewing the developer documentation.

There are also some formal evaluation models you can use to help provide an evaluation framework. Navica's Open Source Maturity Model (OSMM) is published under an open license and assesses the maturity of key project elements. The Business Readiness Rating is a proposed evaluation model under development by a group of major organizations that seeks to expand on both previous OSMM models to develop an open standard for OSS rating.

Biography

Mark Aberdour founded opensourcetesting.org in 2003, which aims to boost the profile of open-source testing tools within the testing industry, principally by providing users with an easy-to-use gateway to information on the wide range of open-source testing tools available.
Mark is Head of Open-Source Solutions at Kineo, a company specializing in online education platforms and content, with a particular focus on the open-source learning management system, Moodle. Prior to this role Mark spent 10 years in software testing, test management and technical project management, and retains a strong passion for software quality.

Agile Testing Days 2011

Nov 14 -17, 2011 in Berlin-Potsdam

The **Agile Testing Days** is the European conference for the worldwide professionals involved in the agile world.

Call for Papers

The Call for Papers ends by midnight on January 31, 2011 (GMT+1).

Please have a look at **www.agiletestingdays.com**!

How open-source tools can make a low budget team perform like the major league?

by Bernard Shai Lelchuk & Dario Uriel Estrin

When you come to think about testing tools, many think about expensive automation and bug tracking tools. However, the open-source community provides numerous tools that can help you set up a lean testing team. You can get everything you need from desktop OS for daily usage to sophisticated testing tools.

Almost any testing need will have a full or partial open-source solution. Even if it is not listed here or your open-source guru friend never heard of it, don't despair, it does not mean it doesn't exist.

You should start your search for open-source tools in two places

- Sourceforge.net – Source forge it the biggest hub for open-source applications.
- Google – well, you know Google already. Just search for „.... open-source testing tool" or „open-source" / „free" +'your testing need'.

The open-source community has its own rules of etiquette. Do not be a leech, be a good community member; anyone can contribute according to their personal skills. Even if you are not a coder, most of the open-source projects will be glad to see you joining the community and contributing by testing or with localization efforts. You could also help by documenting features, innovations and workarounds. You could join the relevant online forums and share with others your insights, making it easier for them to implement the tool and create a body of knowledge that will serve others.

This will be your way to pay back for the common good, assist in improving the tools, get to know this amazing and highly intellectual community and improve your personal knowledge and skills.

We know you are all busy and won't enjoy reading a lengthy article. Thus, we save you the expected charts showing productivity or quality trends jumping sky high if only you used open-source tools. So, here we jump directly to a list of open-source & additional free tools we know well and some we just ran into sometime, accompanied by a short explanation on each.

Operating Systems

When you are a young company or a new testing team, infrastructure expenses can be a big chunk of your budget. Using open-source free infrastructure is a great solution. In the past using Linux distributions was problematic because of various reasons which are no longer issues with today's Linux versions.

CentOS

CentOS is a free Linux version, which is largely based on Red Hat Enterprise Linux, and is thus a good choice for servers' installations. It is a great base for testing environments and installation of testing tools (e.g. Installing Bugzilla, see below)

URL: http://www.centos.org/

Ubuntu

Ubuntu is a natural OS choice for your personal computers. Installation is swift; it has an intuitive GUI and would meet almost all your testers' needs. The usefulness and lightness of Linux OS is enhanced by this well-designed Linux distribution. It will provide you a light and stable operating system.

URL: http://www.ubuntu.com/

Open Office

Open Office provides a very versatile solution of office tools. Its compatibility with Microsoft Office's documents formats has greatly improved, but you should remember it's not perfect. Open Office is a package of applications that provides most of the functionalities you would expect like word processing, spreadsheets and more.

URL: http://www.openoffice.org/

Wiki

Every organization needs to share and document information. There are commercial and expensive tools with tons of features, but they often take lots of time to implement and adapt, require dedicated server hardware and will obviously cost money. Wiki systems (which have numerous flavors) are a relatively simple application, much like Bugzilla is. Simplicity is a common thread amongst many of the open-source tools. A quick and swift installation will provide your team with a tool to share information between team members, with other teams, and for documenting everything from the smallest details up to high level designs and test plans.

URL: Example of Wiki Systems:

- http://www.mediawiki.org/
- http://twiki.org/
- Comparison of Wiki tool: http://www.wikimatrix.org/

Data Syncing Tools

DropBox

The most recognized and utilized sync tool on the market. It's not open-source; however its free version offers 2GB of storage which can be extended to 8GB if you refer more users to DropBox, which is a fair deal. Its client can be installed on multiple machines, thus keeping your data in-sync on any machine of yours. Additionally your data is saved securely on a remote server, thus you can access your files anywhere.

URL: http://www.dropbox.com/

iFolder

Simple & Secure storage solution which enables you to backup, access & manage your files remotely.

URL: http://ifolder.com/ifolder

JFileSync

Java File Synchronization tools for syncing files & folders between different machines, e.g. laptop & desktop.

URL: http://jfilesync.sourceforge.net/ .

SparkleShare

Allows you to host your own service, control your data, share, collaborate and keep record of changes with ease. It's currently released as a Beta for Linux; however, additional Max OS X & Windows versions will be available soon.

URL: http://sparkleshare.org/

Lsyncd

Lsyncd is a Linux syncing daemon which syncs local directories with directories on a remote machine running the rsyncd daemon.

URL: http://code.google.com/p/lsyncd/

Management Tools

Bug Tracking & Project Management Tools

Bugzilla

Bugzilla is THE open-source QA tool. Bugzilla as its name insinuates is a bug tracking tool. We came to work with various defect reporting and tracking tools. Each has its pros and cons, so has Bugzilla. However, Bugzilla is free for any number of users, easy to install, light-weight and creates minimum interference with other server residing applications, which means it doesn't need a dedicated server. In many cases Bugzilla is adopted also as a tool for ticket management (by the IT team for an instance) or even tasks/project management...

- The interface is pretty basic, though some would say unattractive and old-fashioned.
- Good learning curve
- Our Bugzilla has been running for over 6 years with one failure, and that was our fault.
- Portable from Windows to Linux and vice versa using ALMWork's free Virtual Bugzilla Server (http://almworks.com/vbs/overview.html)
- Web based (no client needed)

URL: http://www.bugzilla.org/

Trac

A web based project management & bug-tracking tool with subversion interface and an integrated Wiki.

URL: http://trac.edgewall.org/

Mantis

A web based bug tracking tool with integrated Wiki, chat, RSS Feeds, time tracking & much more.

URL: http://www.mantisbt.org/

RedMine

Flexible project management & issue tracking application with an increasing set of plug-ins, directory, Gant chart & calendar, time tracking functionality, file management & sharing functionalities, per project wiki & forums, repository browser & differences viewer, feeds & e-mail notifications, multi-language support and more.

URL: http://www.redmine.org

OpenProj

A cross-platform project management solution which offers advanced scheduling engine, Gant charts, network diagrams, WBS & RBS charts, earned value costing and more.

URL: http://openproj.org/openproj

Automation & Unit Testing Tools

In the field of IDEs and unit testing tools & frameworks, the open-source community offers the most diverse pool of tools. If you are either a developer or an SQA engineer, you can find a tool for any scripting & programming language, with different levels of functionality, flexibility, robustness and maturity. Below is a list of selected tools in that category, which is only the tip of the complete spectrum of tools. Make sure to visit the following link for a comprehensive list of automation & testing tools:

http://www.opensourcetesting.org/

JUnit

Regression testing framework

URL: http://www.junit.org/

Similar tools: NUnit & TestNG.

JSystem

Functional tests automation framework based on JUnit.

URL: http://www.jsystemtest.org/

Jameleon

Data-Driven Automated testing framework

URL: http://jameleon.sourceforge.net/

Fitnesse

Collaborative Acceptance Testing Framework & documentation tool

URL: http://fitnesse.org/

GenerateData

Open-source data generator script

URL: http://www.generatedata.com/#about

Desktop

Wireshark

Wireshark is the most renowned communication sniffing tool allowing you to look into the internals of the interaction between your application and servers or other applications (e.g. desktop application, mobile simulator communication with servers).

URL: http://www.wireshark.org/

Browsers

Today's internet browsers are important testing tools for two main reasons. First, many products and systems are web based. Second, there are numerous tools and plug-ins for web browsers which facilitate testing via the browser. It is worth mentioning that most of the browser plug-ins are free but aren't open-source – though, for the scope of this article, we decided they fit-in given that they support a lean testing team. Additionally, what is correct for all the tools reviewed in this article is even more correct in this case - there are numerous good tools out there that weren't mentioned, so just go Google them.

FireFox Plug-ins

Automation Tools

- **Selenium IDE** – THE most renowned & versatile web automation tool suite. With the combination of Selenium IDE, RC & Grid you can automate almost every web application from standard HTML based to Rich Java & AJAX web applications. It runs on all platforms (Windows, Mac, Linux, Solaris and others) and Browsers (FireFox 2/3, IE 7/8, Safari 2/3, Opera 8/9), supports many programming languages (C#, Java, Perl, PHP, Python, & Ruby) and can be integrated with most common Testing Frameworks (JUnit, Nunit, TestNG, Bromine, Test::Unit, etc.) and thus can fit virtually **anyone**.

  Moreover it has an increasing knowledge base (SeleniumHQ), an active Google group and a never-ending fan list worldwide. Therefore it's simply reasonable that it has been deployed on many Web Application products.

- **Watir** - A flexible and powerful cross-platform open-source family of Ruby libraries for automating web applications.

  It supports multiple platforms and browsers, uses Ruby scripting language and driven by an active & growing community.

  URL: http://watir.com/

- **iMacros** – A simple and intuitive Web Automation tool for simple repeated daily tasks, as well as compound data scraping and functionality automation.
- **Molybendum** – Web Automation
- **Tellurium** – Web Automation

Client Simulation

- **Modify Headers** – Simulate various mobile clients and user-agents
- **XHTML Mobile** Profile – Add support of XHTML Mobile Profiles

Data validation & extraction

- **Total Validator**- HTML validation
- **Html Validator** – Another HTML validator
- **W3C Page Validator** – A W3C unified markup validator of various web document formats
- **Regular Expression Tester** – Extremely useful for Regular Expression creation and testing
- **LinkChecker** – A simple link validity add-on

Debugging, Analyzing & Data manipulation

- **Firebug** – The most comprehensive web debugging add-on. Its many features include editing, debugging & monitoring of CSS, HTML & JavaScript on any web page.
- **Flash FireBug** – Debug your own SWF files
- **YSlow** – Web performance analyzer by Yahoo! which integrates with Firebug.
- **Fiddler** – Web sniffer
- **Live HTTP Headers** – View HTTP headers of a page while browsing

Security Auditing & Penetration Tools:

- **HackBack** – Post data manipulation & security audit/penetration tool. We found this simple tool to be extremely useful for Client/Server requests manipulation thanks to its URL encoding/decoding & splitting capabilities.
- **Tamper Data** – View/Modify HTTP/S headers & POST parameters.
- **Access Me** – Vulnerabilities test tool using unauthenticated-access requests
- **SQL Inject Me** – Vulnerabilities test tool using SQL injection attacks
- **XSS Me** – Vulnerabilities test tool using cross-site scripting (XSS) attacks

UI & Accessibility Tools

- **FireSizer** – Resize your window dimensions to test how your applications will look in different window sizes.
- **Measure It** – Test height and weight of web elements
- **Accessibar** – Easy manipulation of web page display and text-to-speech output

Productivity

- **FireShot** – Grab detailed screenshots of key areas or a complete (and long) web page.
- **Dummy Lipsum** – Generate „Lorem Ispum" dummy text

IE Plug-ins

- **Fireshot** - An IE version similar to the FF add-on

  URL: http://screenshot-program.com/fireshot/

- **Fiddler** - Web debugging proxy for HTTP traffic / Sniffer

  URL: http://www.fiddler2.com/fiddler2/

- **TamperIE** - Tampering HTTP requests similar to TamperData on FireFox

  URL: http://www.bayden.com/TamperIE/

- **GreaseMonkey** for IE - Similar to the FireFox add-on

  URL: http://www.gm4ie.com/

Load Testing Tools

- **Apache Bench (AB)** – a simple yet powerful http load generating tool for Windows, Mac & Linux

  URL: http://httpd.apache.org/docs/2.0/programs/ab.html

- **Jmeter** – an Open-source functional load test tool which can stimulate heavy load and analyze a server's performance

  URL: http://jakarta.apache.org/jmeter/

- **WebLoad** – A powerful load generator engine for performance and stress testing

  URL: http://www.webload.org/

Funambol

An open-source mobile cloud sync server, which is free for those that support and contribute to its community. This is a very interesting project which offers a complete mobile cloud sync server. Features include device management, push emails and user data and media data and cloud synchronization.

URL: http://www.funambol.com/solutions/opensourceplatform.php

Sites and forums

There are numerous sites and forums with information on testing tools and applications.

See this one for example: http://www.opensourcetesting.org/functional.php

Methodologies

In many organizations the QA department is a leader of innovation and methodology improvements. Methodologies aren't really open-source as they are not applications. Nonetheless, many development methodologies are represented by numerous free online resources that allow you to read about methodologies and techniques and thereby improve your knowledge. Your team will benefit from your knowledge as well as from improved work if you decide to go ahead and push towards implementation. If you are on a slim budget, do not get tempted to spend money on courses – you can read and implement – there will always be time to take a course.

One great example is the Scrum methodology which we implemented in our testing and development teams with great success. There is plenty online reading material, forums, groups in LinkedIn and others.

URLs:

- Google for ‚Scrum'
- http://en.wikipedia.org/wiki/Scrum_%28development%29

Summary – Why open-source?

A testing team with creative, curious and technology-loving team members would benefit from the open-source community on various levels. The team will find a virtually ever-renewing fountain of tools and innovations. Tools are free of charge, thus the only consideration whether to use them or not will be professional and not financial. There is a huge variation. As a team leader you can provide your team members with a curiosity and satisfaction fulfiller, by allowing them to roam and research for new tools – with a specific need in mind or looking for improvements. The latest approach will send a clear sign to your work mates and workers of strong technological orientation and innovation, which would benefit your team internally and externally.

A common pitfall is the fact that not everything that exists should be used, implemented or even take up your time in learning it. Like most of us at a certain point decide on taking a risk by not testing a certain obscure corner of a product, the same rule applies here (as in real life, by the way). You could keep searching for the perfect tool forever, but don't waste time; get the tool that does the work you need, (or most of it), and start using it.

Open-source tools are a professionalism enhancer. If you stick to the above rules you will get a testing environment in which new tools are introduced and used upon demand and not because ‚we already have them', ‚we already learned them' or worst ‚we have already paid for them'.

Your team will gain flexibility, continuous learning and yes, also that important thing called reputation. Your managers will like the fact that things are being done fast, more professionally and of course for free. When a tester does not have to get budget approval for every move, he can move faster, implement any tool upon need matching each feature or products with the relevant testing tools gaining overall enhanced quality. Relations with developers will probably also improve over time, as testers and developers will find points of mutual interest and will share technical know-how and appreciation.

As you move forward, the use of open-source tools will transform your team – both technically and personally. An environment in which you keep introducing new tools upon demand won't easily tolerate team members without the right technical attitude and skills. Your team members will keep learning new things and will not get stalled, preventing frustration, wear out and churn. At the same time, you could leverage open-source tools for enhancing documentation and internal training procedures. Testers should document their findings on tools and the usage techniques so that peers can easily learn the same tools. Internal team instruction sessions are recommended; team members will learn more and faster while the presenting members will gain presentation experience.

Talking of open-source, we could not do without some more ideology infusion. Software is all about people – people are those who will use software as well as those who plan it, code it and test it. It is a strongly human-related industry – you have very few resources, assets or markets which are not people dependent.

We all spend a substantial share of our life at work. As such we strive to forge a community sense. As testers we would also like to be part of a testing community that will support and nurture its members. We are also all members of our cultural and political communities ; what we call citizenship.. Becoming an open-source community member won't require a passport or formal affiliation, but will provide you and your peers with an additional circle of belonging and good citizenship. Social behavior and attitude is all interrelated. Thus, inspiring contributions to this community from your workplace, employees and managers, will eventually improve your work community and your quality of life and reciprocally the quality of your products.

Good bug hunting!

Biography

Dario U. Estrin, Vice President of Operations, WorldMate Inc.
Dario started his career as a programmer in 1994 and was initiated to SQA as a testing automation programmer. He later led testing teams. In WorldMate he was responsible for various departments including SQA, support, IT and corporate sales. As a veteran who worked in various startup companies, Dario brings a wide and integrative view. He strongly believes in the additional value that interdisciplinary testing teams hold and that a strong testing team is a pivotal player in any successful product by integrating business, customer and technological views into one.
Dario is a Certified Scrum Master. He is currently studying for an MBA and holds a B.A. in Political Science. He was a university assistant teacher and led a research team which focused on IT systems in public institutions.
http://il.linkedin.com/in/darioestrin

Bernard is QA Manager at WorldMate Inc., holds a Bachelor in Recording Arts, is a certified System Administrator, a certified Audio Engineer and a passionate musician. His diverse experience in the IT and Music production domains were his key to the testing world, which was followed by many years of beta testing music applications. This know-how allowed him to achieve wide recognition of his testing abilities, subsequently leading testing teams on various platforms and projects.

Bernard joined WorldMate in 2006 where he took his first formal position in the software testing industry. During his career Bernard mentored many novice/veteran testers. He was responsible for most of the company's product testing & release processes, thus obtaining a wealth of knowledge in testing tools, automation, techniques and methodologies .
Bernard published several articles on web and mobile testing. In his free time he is a frequent guest blogger, content coach, community forum moderator and contributor at uTest (the largest on-line professional crowd-sourcing testing community).
http://bernardlelchuk.wordpress.com/

Belgium Testing Days

Testing driven by innovation

February 14 –17, 2011 in Brussels, Belgium

Díaz & Hilterscheid together with **AQIS** is proud to present the Belgium Testing Days from the 14th to the 17th of February 2011, to be held at the Sheraton Brussels Airport Hotel.

These four days will not only be filled with innovative thoughts and exciting ideas by some of the best known personalities of the agile and software tester world.

You as our visitor will also have the opportunity to connect with other QA professionals! In between and after the personally selected tracks, which are a combination of keynotes, presentations and workshops, you will still have ample time for important business networking and significant exchange of ideas.

As a special plus we will offer all registered attendees the exclusive opportunity to speak directly with a **Keynote speaker** of their choice in a private 15 minute **One-on-One meeting**. You can register for these sessions at the conference!

On day 2 and 3 we will have a **Testing-Dojo-Lab** running parallel to the conference. If you are interested in participating in this exciting live-session please register directly at the conference and make sure to bring along your laptop! We grant each person a maximum participation of 2 hours.

Additionally we have added an extra bonus day to the conference. On Day 4, Experimentus will present the **Test Maturity Model Integration (TMMi)**. In 2008, Experimentus was the first company in the UK to be accredited to assess organizations against TMMi.

On the same day we will have a **Work Camp**, preparing interested attendees for the ISTQB Foundation Level Exam and at the same time an **Exam Center**, where you can take your ISTQB Foundation/Advanced Level and IREB Online Exam with additional certification by iSQI.

Register at

www.belgiumtestingdays.com

until December 31st and secure the **Early Bird** registration fee!

We look forward to your participation at the Belgium Testing Days!

Gold Sponsor

ps_testWare
Your devil's advocate

Exhibitors

Atos Origin
CONSULTING > SOLUTIONS > OUTSOURCING

coralius

Díaz Hilterscheid

AQIS

Supporters

testing experience

Agile Record

A Díaz & Hilterscheid Conference / Endorsed by AQIS

Belgium Testing Days

Tutorials
February 14, 2011

- **Johanna Rothman:** "Becoming a Great Test Manager"
- **Lisa Crispin:** "Cover All Your Testing Bases with the Agile Testing Quadrants"
- **Julian Harty:** "Test Automation for Mobile Applications"
- **Stuart Reid:** "Risk-Based Testing"
- **Hans Schaefer:** "A Minimal Test Method for Unit Testing" *

*Afternoon Session only

Conference (Day 1)
February 15, 2011

Time	Track 1	Track 2	Track 3	Exhibitor Track
08:30–09:30	**Conference Opening & Keynote** **Johanna Rothmann:** "Lessons Learned from 20 years of Managing Testing"			
09:30–10:30	**Tim A. Majchrzak:** "Best Practices for Software Testing: Proposing a Culture of Testing"	**Miguel Lopez:** "Using the Machine to Predict Testability"		
10:30–11:00	Break – Visit the Expo			
11:00–12:00	**Susan Windsor:** "How to Create Good Testers"	**Jeroen Boydens, Piet Cordemans & Sillie Van Landschoot:** "Test-Driven Development in the Embedded World"	**TestLab** **Markus Gaertner:** "Martial Arts in Testing? – Testing and Coding Dojos"	
12:00–13:00	Lunch – Visit the Expo			
13:00–14:00	**Steve Caulier & Erwin Bogaerts:** "Above and Beyond the Call of Duty"	**John Bertens & Remco Oostelaar:** "The Business in the Driver Seat with Cloud Computing/ How to Test – A Dialogue Between the Old and New Generation Tester"		
14:00–15:00	**Peter Morgan:** "Poor Defects BUG Me"	**Daniël Maslyn:** "Virtualized Testing: Opening the Doors of Opportunity"		
15:00–15:30	Break – Visit the Expo			
15:30–16:30	**Keynote** **Stuart Reid:** "Innovations in Software Testing: the Past, the Present and the Future"			
16:30–17:30	**Lightning talk** "Looking into the Future"			
17:30–18:30	Conference Reception – Networking in the EXPO Hall			
18:30	Show			

Belgium Testing Days

Conference (Day 2)
February 16, 2011

Time	Track 1	Track 2	Track 3	Exhibitor Track
07:15–08:30	Bonus: Surprise Breakfast Sessions			
08:30–09:30	**Keynote** **Julian Harty:** "Alternatives to Testing"			
09:30–10:30	**Nathalie Rooseboom de Vries van Delft:** "Unusual Testing: Lessons Learned From Being a Casualty Simulation Victim"	**Rocio Guerra-Noguero:** "How to Successfully Test in the Land of Babel: A Case Study"	**TestLab** **Markus Gaertner:** "Martial Arts in Testing? – Testing and Coding Dojos"	
10:30–11:00	Break – Visit the Expo			
11:00–12:00	**Bjorn Boisschot:** "The A(utomation)-Team"	**Jamie Dobson & Jorrit-Jaap de Jong:** "The Fifth Dimension: You've Just Entered The Financial Engineering Zone"		
12:00–13:00	Lunch – Visit the Expo			
13:00–14:00	**Keynote** TBD			
14:00–15:00	**Anderson dos Santos & Bruno de Paula Kinoshita:** "How to Automate Tests Using TestLink and Hudson"	**Gojko Adzic:** "Winning Big with Agile Acceptance Testing – Lessons Learned From 50 Successful Projects"	**TestLab** **Markus Gaertner:** "Martial Arts in Testing? – Testing and Coding Dojos"	
15:00–15:30	Break – Visit the Expo			
15:30–16:30	**Graham Thomas:** "How to Suspend Testing and Still Succeed – A True Story"	**Jurian van de Laar:** "How We Bridged the Gap Between Developers and Testers in a Scrum Context"		
16:30–17:30	**Keynote** **Lisa Crispin:** "Learning for Testers"			
17:30–17:45	Closing Words & Awards			

Bonus Sessions (Day 3)
February 17, 2011

Time	Track 1	Track 2
09:00–16:00	TMMi by **Experimentus**	Work Camp: Preperation for Exams – ISTQB Foundation Level with **Werner Lieblang** Exam Center: ISTQB Foundation and Advanced Level, IREB Online Exams – Certification by iSQI

Please visit **www.belgiumtestingdays.com** for the current program.

Does any one know a good open source tool for Test Management and Bug Tracking?

by Vinodh Velusamy

This is a discussion topic started by senior test engineer Chakradhara Pernumarthi on the *Software Testing & Quality Assurance* group in Linkedin; one of the most popular business-oriented social networking websites. This discussion is one of the most popular currently on the group discussion board, with 120 comments from group members, which primarily consisted of software testing and quality assurance professionals. The discussion was started in April 2010, probably inspired by another popular discussion in the same group '*Bug Tracking Software - What's your tool of choice?*' started by QA analyst Domonic Prince. Domonic had started his discussion in October 2008, and had generated about 260 comments, before mellowing out in March 2010.

Both these discussions have given a unique opportunity to gauge the popularity of tools, both open source and commercial ones, among the testing community. They can also be considered as a great resource for valuable feedback on what the specialists in the field look for in such tools when trying to choose one.

This article provides a roundup of the most popular tools that anyone evaluating bug tracking and test management software should consider with insight and tips from the experiences of testing professionals.

Introduction

Although open source software tools will be the primary topic of discussion, commercial tools will also be discussed in this article. First, the list of open source software tools that were recommended by the members of the group in Chakradhara's discussion on open source software will be discussed. Then we would like to look at the software recommendations from Domonic's discussion, which included non-open source as well. The reason for this is that most people evaluating bug tracking and test management software are going to be looking for feature sets and functionality that are available in commercial tools. Hence we will also discuss what the most popular commercial software is, what they offer, and why they are considered for test management and bug tracking.

If we can consider the discussions as a survey, figure-1 shows the popularity of the different commercial tools available, including commercial and open source software.

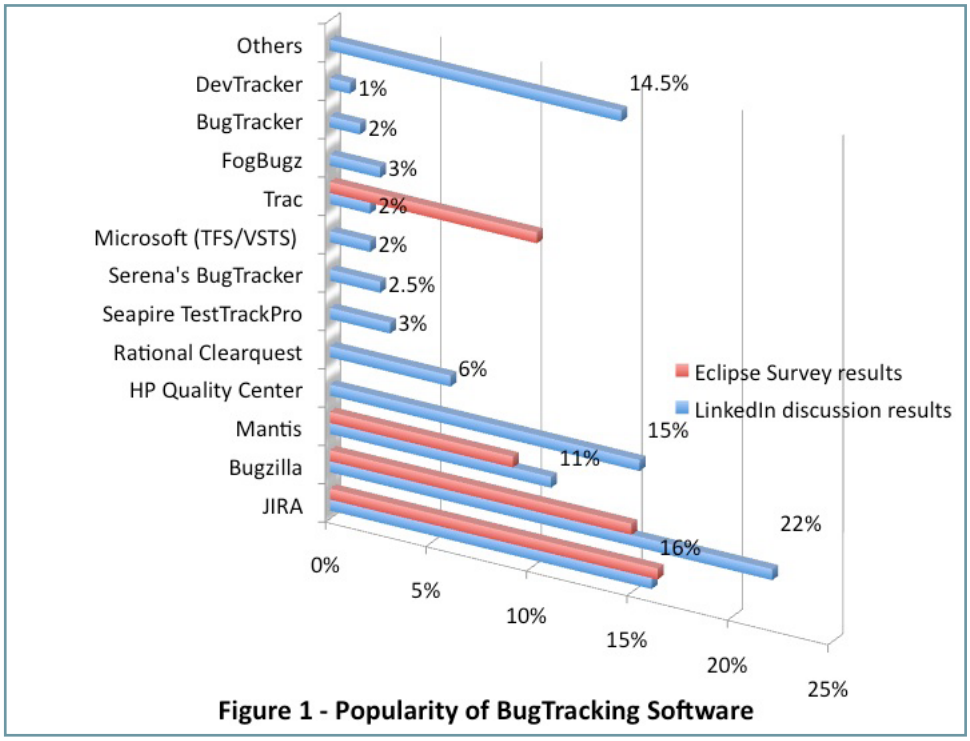


Figure 1 - Popularity of BugTracking Software

The percentages inferred from the discussions seem to match the results published from survey *Open Source Developer Report 2010* conducted by the *Eclipse community*. Atlassian's JIRA software was popular among 16% of both the *Eclipse community* report and linkedin group members. TRAC was popular among 10% in the *Eclipse community* report while only 2% of the members preferred it in linkedin group discussions. In the discussion, Mantis was popular among 11%, which was quite close to the 9% obser-

ved in the *Eclipse community* report.

The most popular bug-tracking tool that members of the group preferred was Bugzilla, attracting 22% of them. 15% of the members preferred HP's Quality Center (Formerly Mercury Test Director), while 6 % liked Rational ClearQuest. Microsoft Team Foundation (TFS/VSTS) and FogBugz were each popular among 2% of the linked in members. 3% liked Seapine TestTrack Pro.

We shall now see what the members liked and disliked about the various products, which we shall broadly classify under commercial software and open source software.

Open Source Bug Tracking Software

The figure-2 shows the popularity of the different open source bug tracking software tools discussed in the linkedin group discussion by the members.

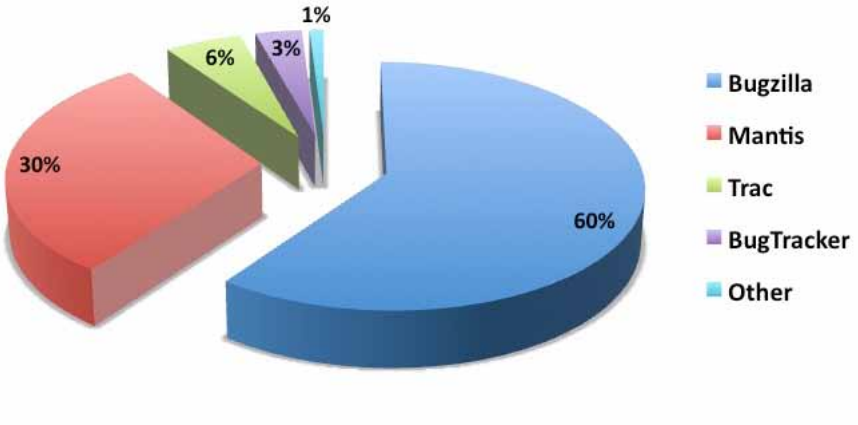


Figure 2 - Popularity of opensource software

Bugzilla

Bugzilla was the most preferred bug tracking software among the members of the discussion group convincing 60% that it is the most ideal open source software.

Jack Mangan says that I know Bugzilla isn't perfect, but it does the trick. There are a lot of customizable reporting features and its UI isn't too difficult for the non-techs to navigate. Nice price too.

Amjad Kayani however thinks that although it falls short in terms of reporting capabilities, it can work fine for smaller projects.

Daniel Emerald has a great tip that there is a Windows-like overlay called Deskzilla that they are starting to use for those that want that form of UI.

Although it is the most popular tool, there were some members like David Baer who found Bugzilla is by far the worst. He thinks that setting up bug flow rules and management of individual bugs is very cumbersome. He also found searching for bugs was unnecessarily complicated an ineffective.

Akashdeep Awasthi summed up the features generally considered by most members when they want a bug tracking tool. He says; *if you really want to go open source for your bug tracking and test mgmt, then no doubt „BugZilla" is the best one to go, due to following reasons:*

1. Easy to setup and easy to understand.
2. Easily integrates with TestLink which is a test management tool.
3. Very test friendly.
4. Contains almost all the features found in commercial products.

Rupali More suggested that Bugzilla and Test Runner or Bugzilla and Testopia can be used for Bug tracking and test management. Integration of Bugzilla and Testopia works really well. Bugs can be mapped to the test cases directly and it becomes easy to track your work.

Mantis

Mantis was the next popular open source bug tracking software, preferred by almost 30% of the members of the discussion group.

Shripal Shah says that Mantis is and open source tool with enough user-friendly functions having capability to engage any time of project size and bug count. Apart from other tools available online, Mantis can be hosted on your own server with very little know-how of PHP and MySql. He has found that Mantis can be made operational within no time and time to learn is very low. With custom field creation facility, he has also been able to use the tool as a project management tool for some of the mid-sized projects. He sees no reason in spending a lot of money if you get everything to track and monitor bugs using open-source tools like Mantis.

Todd Lawall considers Mantis' clean web based interface, ease of setup, minimum resource requirement (could run on an obsolete desktop if needed) as a big plus. He also points out that it doesn't require any particular platform for the client, only a browser, and no fancy scripting, so even smartphones can handle it.

Rajat Prakash thinks that Mantis is good for the testers as it provides a simple enough interface for defect entry. But he feels that from a manager's perspective JIRA is much better as it provides the summaries, trends and reports that are needed to manage a project effectively.

Francis Law summed up the Bugzilla vs Mantis debate thus, Both Bugzilla and Mantis are very popular, but none of them has a clear win over the other. Some people comment Bugzilla is very powerful but difficult to setup and customize. Some people comment Mantis is not robust enough. I use Mantis and like its ease of setup/use and ease to customize. If you are a single person team, you can even use the instantMantis that can be setup and ready to use within 15 minutes.

TRAC

Trac was a popular choice of open source bug tracking tool among 5.5% of the members.

Jason Galyon comments that if nothing else he enjoys its great source viewing (changesets, diffs, comparisons between arbitrary changesets, etc.). He thinks that Trac is extremely configurable and extensible and the source control integration is fantastic. He

uses it as a primary wiki and exports to Excel, Word, and Open Office as needed.

Hugues Bonin has used DDTS, Rational ClearQuest and Trac. His favorite of all is Trac: great for small projects, allow tasks tracking as well and it's FREE (open source). The community helps improving it with plug-ins.

Sumit Madan thinks that Trac helps to get the developers involved. It integrates easily with subversion and the IDE of the developer. This way the developer can have a Trac plug-in while checking in fixes and can add their comments. This helps the QA team link directly to the code changed in the bug comments to see how invasive the change (and hence how extensive the testing) is. It is all coded up easily in Python and super configurable - also not as heavy as Bugzilla or cumbersome to configure as Mantis.

Enrico Oliva asks why not use Test Manager plug-in for Trac to integrate test management.

Robert Evans explains that they use TestLink + Trac. Integrating TestLink as a test management tool was a good choice for their needs.

BugTracker

The open source bug-tracking tool BugTracker was recommended by about 3% of the members.

Pankaj Goenka recommends the web-based bug tracker as it provides features like high configurability, email integration, search, screen-capture utility and ability to add custom fields.

Sreenadh OG further explains that after a review of numerous available solutions he preferred BugTracker. According to him it is the only BugTracking tool written by Tester from Tester's (Bug Tracking tool user's) perspective I feel. He says, *Use it, review it and arrive at your own conclusion and please don't get mislead by the not-so good looks of the poor demo hosted in their site (this bug tracker is highly customizable and it is very easy to change its looks and feel by applying a simple custom CSS). The awesome fact is that the whole product is developed by a single individual, and his future target list and future plans are also simply unique and herculean!*

Open Source Test Management and Complete Solution Software

Figure-3 shows the popularity of the different open source test management software tools and open source tools that provide complete solutions containing bug tracking and test management, that have been recommended in the linkedin group discussion by the members.

TestLink

As an open source test management tool, TestLink had the maximum recommendation at 53%.

Like Srikanth Reddy Komandla , most found it easy to create and manage Test cases as well as organize them into Test plans using TestLink. These Test plans allow team members to execute Test cases and track test results dynamically, generate reports, trace software requirements, prioritize and assign tasks.

The general consensus was, as Renuka Arunkumar points out that it possesses almost all qualities of HP Quality Center. It needs to be combined with a defect-tracking tool, like Mantis, Bugzilla. Renuka uses Redmine. Francis Law had a good tip that it even includes a very simple requirement management module, thus bringing it closer to QC standards.

Nitesh Agrawal, Rewati Potdar, and many others suggest that TestLink and Mantis are a good combination.

Testopia

At 17% Testopia is the next most popular open source test management software.

Oana Casapu recommends Bugzilla with Testopia plug-in/extension (for test management). Together they look like HP's Quality Center, although the reporting part doesn't look that nice as in QC, but the most important features are present.

One disadvantage that she sees with Bugzilla+Testopia is that it's not that intuitive for a person that is not involved in software development activity. Business people or clients that just want to see the testing progress may find it a little bit complicated to navigate. But still, she feels, this combination is the best open source test management and defect tracking tool.

Ameer Khoja agrees with Oana, but he feels that installation can be a little tricky. There is a reporting plug-in, which you can install on top of it.

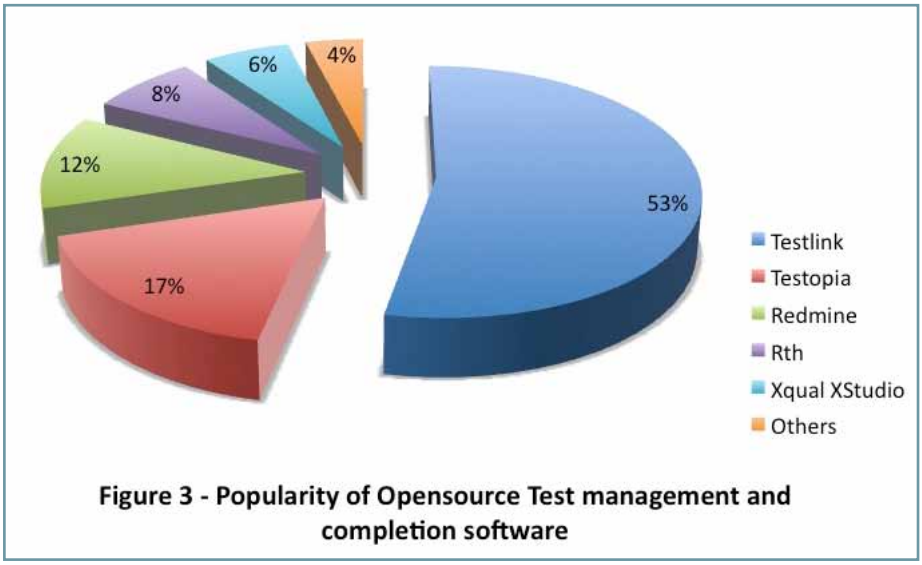


Figure 3 - Popularity of Opensource Test management and completion software

Sarath Chandrasekhar thinks that Testopia, which is basically TestRunner integrated with Bugzilla, is the platform that comes very close to HP Quality Center in terms of documenting, organising, assigning and executing Test cases. It includes a Test Cycle History and a loosely coupled integration between the Defect number and Test case.

Redmine

Redmine is the next popular open-source tool at 12%.

Kiran Deram explains that it was developed on ruby on rails technology. He feels

that it is a wonderful tool for Test Management and bug tracking as it can be customized.

Valmik Darade thinks that it's just like JIRA.

RTH

At 7.5% popularity, RTH is the next recommended open source complete solution software.

As Rizwan Jafri puts it, RTH-Turbo is a simple, open source, light weight, easy to use and learn Test Case management tool that supports Test Case to Requirements, Requirement to Release, Test Case to bug associations features.

Xqual XStudio

Xqual's Xstudio is one of the newer tools that have gained the interest of about 6% of the members.

Eric Gavaldo gives a detailed analysis of its features. He comments that XStudio handles the complete life cycle of the projects: users, requirements, specifications, projects (dev, testing, etc.), tests, test plans, test reports, campaigns defects.

It allows scheduling or running fully automated or manual test campaigns. It also provides some tools to measure the coverage of testing (i.e. traceability matrix); all this via a local installation or through the browser.

It can run manual tests but also automated ones (through the provided open-source launchers): QTP, AutoIt, Selenium, Ranorex, Sahi, TestComplete, TestPartner, Visual Studio, Beanshell, Squish, NUnit, PyUnit, JUnit, TestNG, Marathon, Perl, Tcl, simple executables, bat files, etc.

In addition, the SDK allows one to develop specific launchers to interface with proprietary tests. It includes a very good and appreciated bug-tracking database but can also integrate with: Mantis, Bugzilla, Trac, Jira.

Commercial Software for Bug Tracking and Test Management

The figure-4 below shows the popularity of the different commercial software tools discussed in the group discussion by the members of the business-oriented social network LinkedIn.

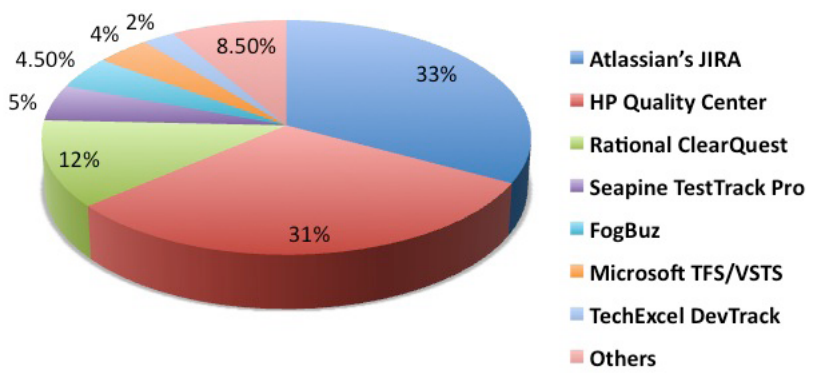


Figure 4 - Popularity of Commercial Software

Atlassian's JIRA

Among the different software tools discussed, Atlassian's JIRA was the most popular commercial software for Bug Tracking and Test Management. Amidst the commercial tools, 33% of the members preferred JIRA.

Most of the members echoed Ofer Prat's comment that he enjoyed JIRA a lot because it is extremely usable, flexible productive tool, easy to maintain and customize, and least expensive. Generally, the consensus was that Atlassian is developing quite an impressive portfolio.

Many of the members, like Jim Weeks, highly recommended JIRA, since their view on bug tracking systems was two-fold. Firstly, it gives them the information they want to see and secondly it is easy and intuitive for QA engineers and development engineers to work with. He says, *I've yet to have a user complain that JIRA gets in their way.*

Some members like Mike Bresnahan, mention that JIRA works well when you have many needs. He says that his company uses it for facilities, tech support, IT helpdesk, hardware engineering, software engineering, and RMA reporting and metrics. The workflows are user customizable and performance is pretty good. The only negative is that the search facility is not very good.

Some of them are so thrilled with JIRA, like Nitesh Agrawal, that they find the best tool ever for Bug tracking is JIRA. He shares his experience thus; *I would say, one should try this tool at least once; it's freely available demo version. It is an excellent and very easy to customize. In the mobile domain, more bug fields are required to define a bug properly. We have customized it successfully and we are executing release management and defect prevention nicely.*

HP Quality Center

HP's Quality Center was recommended by 31% of the linkedin group members for commercial tools. Most of these members were of the same opinion as Kiril Strax, who thinks it is difficult to find another product that would match all the possibilities of QC (e.g. installation/configuration options, very versatile and easy customization, report/export system, etc.). It also scales very well, so overall it's a better match when flexibility is a must, when you want a test/defect integrated solution, or large number of end users must be supported. But it has two drawbacks: the price, and the fact that the client side must reside on Windows (even worse: you are mostly limited to IE).

Members like Galith Nadbornik (Chriqui) enjoy the Quality Center features. Starting from the requirement definition and the linking possibility between the requirements and raised bugs and created Test cases, it simplifies the follow up for testers and provides extensive reporting tools for management. Easy to use, very flexible and the Admin can create as many modules and classification are other strengths.

Another member, Karen Keirstead adds that you can tie everything together from test conditions through defect resolution with all the notes and the dates – time, and person entering the information. etc. It makes the need for traceability matrix documents obsolete. Jonas Stenman echoes this by pointing out that the possibility of tracing

Can agile be certified?

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

Training Concept

All Days: Daily Scrum and Soft Skills Assessment

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

Day 2: Planning and Requirements

Day 3: Testing and Retrospectives

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

Day 5: Practical Assessment and Written Exam

Certified Agile Tester

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

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

Supported by

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

between test cases, defect and requirements are a great benefit. Also the metrics part is very helpful.

Danny Hemminga summarizes the key features. It gives you grip and control on the whole IT and application lifecycle. From setting up your requirements, set up your test cases, test execution, adding defects and link them all at the same time. For larger projects it gives you more than defect based testing, but gives you also the workflow in which every employee has to work. Results, history, root cause, test coverage, defects linked to test sets and requirements.

Some members like John Hantak did have some issues with Quality Center. He shares his experience thus; *Approximately two months ago, my team started piloting HP Quality Center. So far, I've been impressed. The performance is more than acceptable and reporting mechanism covers most of my needs. So far, the only areas that my team is struggling with are:*

1. There is no interested party field in defects created in Quality Center, so you have to manually enter in the names of team members that you want to notify regarding a defect that isn't assigned or didn't create the defect.
2. User A can overwrite user B's comments.
3. The reporting mechanism can only report on two dimensions (i.e. Functional Area versus Priority or Functional Area versus Status). The tool that my team used prior allowed you to report on more than three dimensions (i.e. the Y axis has the Functional Area while the X axis has the different Status/Priority combinations).

Rational ClearQuest

About 12% of the members recommended Rational's ClearQuest as the tool of choice among the commercial ones.

The tool was recommended by members like Sripad Raj because it is famous for user friendly UI (user interface) and multiple features which makes it most suitable for all types of projects.

Seapine TestTrack Pro

Amidst the commercial software, Seapine's TestTrack Pro was preferred by 5% of the members.

Group member Stacy Lukaskawcez shared her experience; *TestTrack links to Seapine's source control (SurroundSCM) and several other third party source control tools but with SurroundSCM you get a bidirectional link that allow defects to link to source code changes for complete traceability.*

Another group member Susan Sutherland Pina, liked Seapine's test track – since it was easy to modify workflow, status, and various other fields to fit your processes and lingo. Reports were decent and they never had an issue with response time.

FogBugz

Among the commercial software, FogBugz brought 4.5% of the member recommendation.

Sharon Matthews shared her experience as; I've used FogBugz for projects both small (150 bugs or less) and large 20,000+) and highly recommend it. Easy to use, you can get email updates as changes are made to each case, great customization abilities, filters, wiki's, reports, project planning, has great screen capture tools (screenshot tool for testers, SnapABug for customers, TechSmith SnagIt for advance screen capture, FreshLog for agile project collaboration, and Bug Shooting for email or Skype), has a host of plug-ins, time management and tracking tools, and then a great interface for entering and tracking bugs.

It isn't too expensive, depending on whether you host it on your server or FogBugz server and how many users you need it for. Take a look at their website or ask me further--I've used it a lot on a wide variety of projects.

Kirsten Anderson recommends that you should go for Fogbugz primarily if you are working in an agile environment. One of the advantages is that it is highly configurable.

Microsoft TFS/VSTS

Microsoft could only attract about 4% of the group member's preference, among the commercial tools.

David Silvan was one of the strongest supporters of Microsoft's tool. He had very detailed comments on the advantages and disadvantages of using this tool as follows:

After using a variety of different bug tracking systems, my personal favorite is Team System (TFS/VSTS). As a stand-alone bug tracking system it isn't anything spectacular, but the real exciting part is how well it integrates with all the other parts of the software development process. Its tight integration with programming, builds, and testing, make it easy to make its use an intuitive part of any software process. For example, a build automatically triggered as part of continuous integration can automatically generate a bug based on failed automatic regression tests. This kind of integration saves a whole lot of work and involves developers much more in the test processes. Another example of tight integration is that bugs can be linked directly to specific lines of code changes, since the code's version tracking is kept in the same system. This is very handy for developers.

While systems like JIRA are also great for including bug tracking as part of an overall process control system, he hasn't seen them integrate as completely as Team System does.

As someone who had designed processes, this made a big impression on our choice when evaluating a dozen different systems. He found the system flexible and customizable enough to be able to be intuitively included in teams using waterfall, agile, SCRUM, and a combination of the above.

The one area they found lacking was an intuitive interface for customer support. But Microsoft came out with a scaled down web portal that worked really well, since they could limit them to not having all the other distracting features and keep them on just bug reports and individual requests.

Team Foundation Server 2010 (TFS/VSTS) does integrate with the full software development lifecycle, including project requirements (i.e. user stories), test plans, test cases, work items, source control, bug tracking, reports, continuous integration builds, workflow management, etc. all integrated in a way that is incredibly powerful.

In other words, when testers investigate a bug, they can drill down to not only exact lines of code that have been changed in code churn, but also see the test cases and requirements that drove the change. Developers, when investigating the bug can see

play-backs of exactly what the tester did that brought up the bug, plus captures of any exceptions and errors. Project managers can get detailed reports of things like bug churn, burn down charts, etc.

TechExcel DevTrack

Unlike Microsoft, relatively unknown tool DevTrack was able to attract about 2% of the group member's preference, among the commercial tools.

Lou Pedron recommended it as a solid product with many features including both a Windows and Web-based GUI. According to him, projects are easily customizable for any application, in minutes. It also integrates with version control systems as well as DevTest for test case tracking. Service & Support are apparently excellent.

David Baer cautioned that DevTrack is a great application but doesn't provide a lot of value over other products if you don't combine it with other tools from TechExcel like DevTest. If you have a need to track test cases/suites and would like to tie them in with bug tracking software (and have the budget for it) then DevTrack with DevTest is a good solution.

Comparison of different software solutions

Jose Endara observes that US Government projects for some reason use (in general) Rational ClearQuest, and Commercial projects HP-Mercury Quality Center.

Mary LeMieux-Ruibal comments; *I'm another one of the JIRA votes. It was a tough conversion since I had been a Mercury automated tool specialist for many years and heavily utilized Quality Center.*

When completing a return-on-investment analysis on issue tracking software for my current client site, JIRA surprisingly came out leagues ahead of the competition. The benefits:

- *Customizable workflows, issue types, screen schemes and mandatory fields*
- *Open source*
- *Easy to use / low training overhead*
- *Cost for an unlimited license is a fraction of the cost of other issue tracking tools charge*
- *The plug-ins for ‚fisheye' for code traceability and the ability to link into Confluence have eliminated the need to keep a separate requirements trace matrix*

My current client site has 875 separate projects running on a single instance of JIRA. We have over 9,000 concurrent users, 300,000 logged issues and the maintenance of JIRA is divided between a team of four persons. I admit we have faced a lot of issues due to the sheer size of our implementation but I cannot imagine how Quality Center or ClearQuest would hold up to the abuse we've put JIRA through. (Not to mention how much 9,000 concurrent users in Quality Center or ClearQuest would cost!)

Carolyn Ford opines that you need a tool that can incorporate the following: Requirements, Project Planning, Defect tracking, Test Planning, Test execution, and metrics to report on those areas in detail. You also need a tool that as „Couples" these areas together, like for Requirements Coverage, Test cases associated with defects, and test coverage. *From a cheapware (Not freeware) option, JIRA offers both defect tracking, project management, issue management all tied into one tool so you can have requirements traceability. HP QC is great if business analysts use the requirements portion and testers use the defect portion, but it does not have a project-planning portion. Defect tracking is good in Bugzilla, but test planning is a miss there. Rational's suite of goodies, great if the organization has the bucks, same with HP. TestTrack pro also misses in the Test planning department.*

Vlad Olteanu comments that the choice is clearly Bugzilla if you are aiming for a freeware solution. The table reporting system is good, and it's easy to read and administer. Add Testopia for project management capabilities. If you are willing to pay and you need to manage/overview a lot of projects/project sections, go for JIRA. A correctly installed and configured JIRA instance will allow you to see everything you need in just a simple look at your dashboard. He has also used Mantis and TRAC, but preferred Bugzilla over both in terms of functionality, referring to how easy it is to overview the entire project, see how debugging and bug reporting go along.

Marjie Carmen would first recommend that you understand the needs of the system, like who needs the information and in what format. What types of reports will be useful for each of the stakeholders internally and if you have to provide reports to people outside of the company i.e. auditors, clients.

He chose Bugzilla when creating a new QA department, as it was open source and scaled well. He has used Rational, HP and Compuware solutions, but feels they were rather cumbersome and expensive for small - mid-size firms. Ideal only if you're going to implement other tools they have utilizing the shared business knowledge across the tools. He feels that JIRA has amazing configuration and reporting functions but still, as with any system, you have to be committed to fitting it into your culture. They needed even a full time JIRA administrator as they continued to grow the tool for uses outside of defect/change management.

Conclusion

William Shaffer has summarized the list of bug tracking tools discussed in the group. The list is at http://waysysweb.com/qa/bugtracking.html

Caroline Ford recommended the link http://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems for comparison of various software tools available for bug tracking.

Considering all the different tools and features, selecting the best bug-tracking tool is an interesting process. One way to go about choosing bug-tracking software is to first list the features you need and prioritize them. If you already have an experienced software engineering team, borrow from their experience on listing the tools and/or features they would like to have in a bug-tracking tool. Consult with other stake holders if you need an integrated solution for requirement gathering, project management, and so on. Discuss with management what type of reports and metrics are essential.

Consider the tools that meet the requirements, consider whether the tool can be customized to suit your needs and then match them with the budget that you have. Explore the possibility of trying out the tools you have shortlisted to evaluate the weaknesses, and revisit your requirements and shortlisted tools.

HP Quality Center is one of the most revered tools that most software testing professionals deem as a standard for test manage-

ment and defect tracking. When anyone is evaluating such a tool, they want to compare it with QC, since it effectively combines both.

Although there doesn't seem to be a single popular open source alternative to QC, there are different tools available for defect tracking that can be combined with test management tools to meet all QC features. But one key issue is that such combinations bring considerable overheads with having to support and maintain multiple application systems. All these need to be weighed in before making a final decision on which tool is right.

References

Linkedin Software Testing & Quality Assurance Group Discussion: Bug Tracking Software - What's your tool of choice?, Started by Domonic Prince [Online] October 2008. [Cited November 10, 2010] http://www.linkedin.com/groupAnswers?viewQuestionAndAnswers=&gid=55636&discussionID=352708&split_page=13

Linkedin Software Testing & Quality Assurance Group Discussion: Does any one know a good open source tool for Test Management and Bug Tracking?, Started by Chakradhara Pernumarthi [Online] – April 2010. [Cited November 10, 2010] http://www.linkedin.com/groupItem?view=&gid=-55636&type=member&item=16264217&qid=c823cee7-a64a-4588-9d7d-5c1cad7c437d&goback=.gde_55636_member_352708.gmp_55636

Open Source Developer Report 2010, Eclipse community - http://www.eclipse.org/org/community_survey/Eclipse_Survey_2010_Report.pdf

Comparison of issue-tracking systems. Wikipedia [Online] November 9, 2010. [Cited November 10, 2010] http://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

Bug Tracking Software, William Shaffer [Online] March 2010. [Cited November 10, 2010] http://waysysweb.com/qa/bugtracking.html

Biography

Vinodh Velusamy is an ISTQB certified professional currently working as Quality Assurance Specialist at Verizon's Security Solution Group. He has experience in software testing for more than 10 years and has been involved in software development for more than 15 years. He has authored articles on open-source software tools in online tech websites such as freshmeat.net. He has also presented at software testing conferences such as Software and Systems Quality Conference, Dusseldorf, Germany and ICSSEA Workshop on System Testing and Validation, Paris, France.

Knowledge Transfer – The Trainer Excellence Guild

Díaz Hilterscheid

Rapid Software Testing
by Michael Bolton

- Jan 10 – 12, 2011 Barcelona
- Mar 16 – 18, 2011 Berlin
- May 23 – 25, 2011 Helsinki

An Agile Approach to Program Management
by Johanna Rothman

- Feb 17 – 18, 2011 Stockholm
- Sep 12 – 13, 2011 Berlin
- Sep 15 – 16, 2011 Amsterdam

From User Stories to Acceptance Tests
by Gojko Adzic

- Jan 10 – 12, 2011 Amsterdam
- Mar 7 – 9, 2011 Berlin

Testing on Agile Projects: A RoadMap for Success
by Janet Gregory

- Feb 7 – 9, 2011 Zürich

Foundations of Agile Software Development
by Jennitta Andrea

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

Risk-Based Testing
by Hans Schaefer

- Feb 17, 2011 Amsterdam
- Jun 10, 2011 Kopenhagen

Website:
www.testingexperience.com/knowledge_transfer.php

Automated tests in Continuous Integration environments.

A case study on open-source tools.

by Zbyszek Moćkun

The aim of this article is to share our experience in building and managing Continuous Integration environments on the basis of open-source tools like Hudson and Selenium. In this article we will concentrate on testing purposes, suggest just few improvements and describe our experience with using open-source tools. The main idea is to present how to use automated tests reasonably by minimizing the time spent on them while optimizing the benefits that automated tests give us. Of course, most of us in the Agile world concentrate mostly on a methodologies based on short iterations.

Tutorials for the tools mentioned are out of the scope in this article, because this would be too extensive to describe here.

Short iterations should finish with demo/release to the client and are based on continuous functional testing. Testers spend their whole time testing new functionality for the iteration, which means there is no time for regression testing. If we add regression tests at the end of a short iteration, we achieve a mini waterfall process.

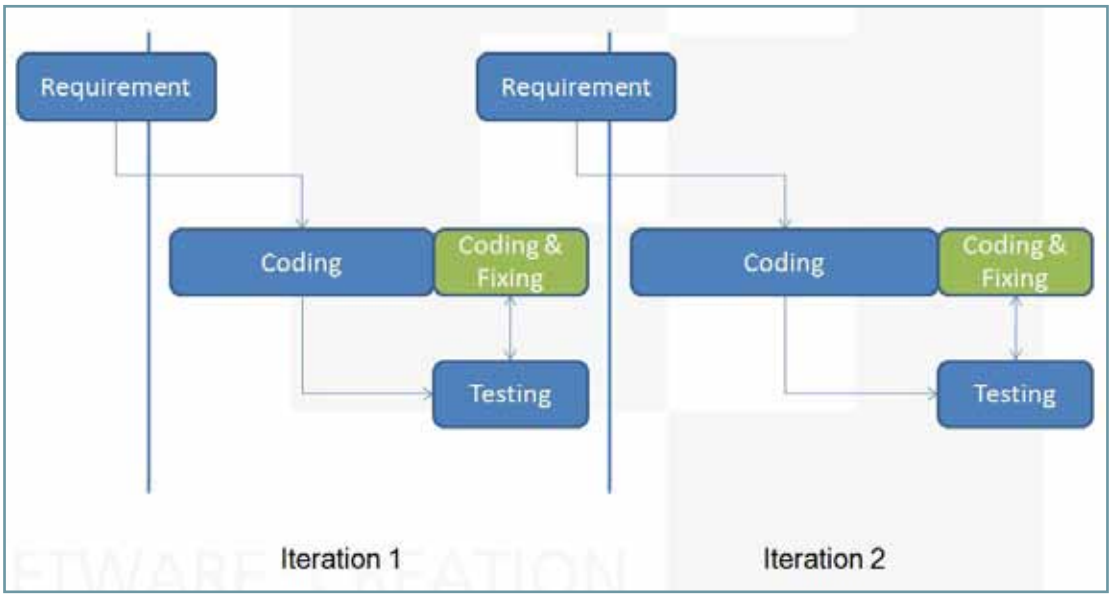


Figure 1 – Mini waterfall

In addition, we lose time from the short iteration. For example, if we have a 10-day iteration and plan regression testing for 2 days, we have only 8 days left. If we add half a day for planning at the beginning of the iteration and half a day for demo, then we have only 7 days left. In conclusion, the regression test at the end of the process lessens the number of delivered functionalities. The following two schemas illustrate this idea.

So how can we have our cake and eat it? Be Agile, use short iterations, deliver high-quality software and don't go back to the waterfall?

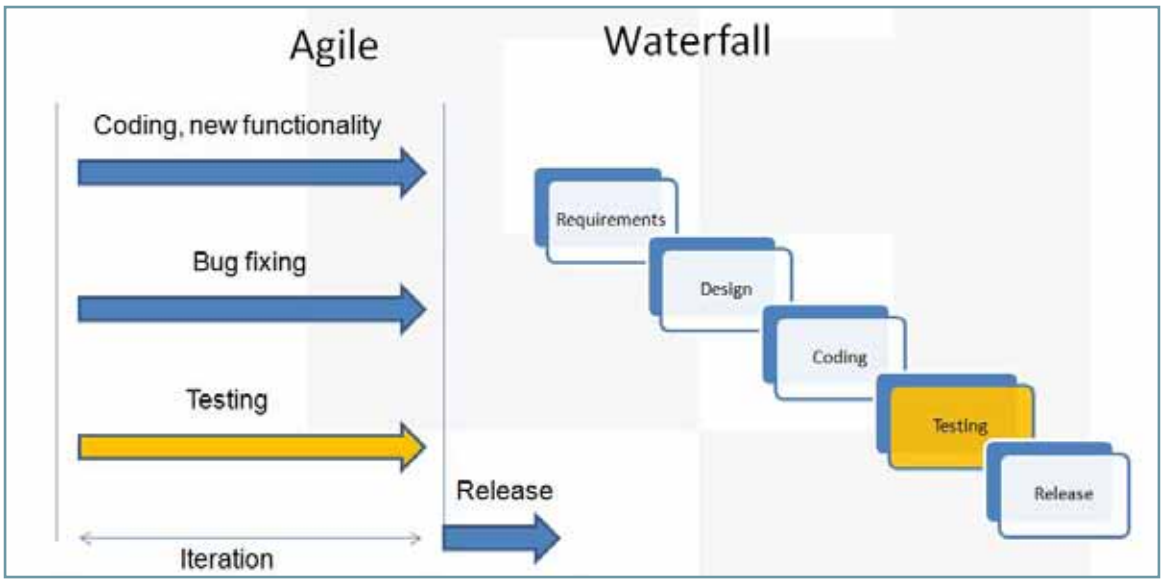


Figure 2 - Agile vs Waterfall

What is CI (Continuous Integration)?

The following quotation I like the best:

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build ***(including test)*** *to detect integration errors as quickly as possible."*

Mark Fowler

This short quotation says everything; The idea is to integrate new pieces of software as fast as possible. Very important here are time frames. Integration should be divided into three steps:

1. Commit new functionality and build new application
2. Run unit tests
3. Run integration and system tests

The first two steps should be run for each commit. They are very specific and shouldn't take more than several minutes. For smaller applications it shouldn't take even a minute. As we can see, running these steps for every commit is not an issue if we talk about time or cost. These two steps can be done without a test (QA) engineer.

The third step, integration and system test, is more complicated and time consume. Usually system tests need more time and require a test engineer.

There are several open-source tools on the market that allow you to introduce continuous integration. Hudson is used in this article as an example.

Figure 3 - Continuous Integration tools

How important is to integrate automated tests with the continuous integration environment?

Agile methodologies are based mostly on short iterations (one or two weeks). The questions are: How efficient is this? How can we be sure that the released software is of high quality? How can we manage and plan tests that fit our budget and resources?

If we think about automated tests – the question is: How often and when should we run automated tests?

Our initial answer is really easy: As often as possible and as quickly as possible.

Both answers suggest one solution: Continuous Integration (CI). CI allows us to run automated tests after each commit and send feedback with results to developers.

Good automated tests should cover all functionality or at least most of it. If we plan to run automated tests at the end of an iteration, there would probably be no time left for fixing any found issues. Not all issues are quick to fix; some need changes that can influence already implemented features or may require a change to the technical solution. What if a new feature has a bigger impact on the application than we thought? There is no time for changing the new feature or asking the client for additional feedback. All of this suggests that regression tests should be done during the whole iteration. The question is how to use automated tests as often as possible without causing a big effort for the QA team when they should be concentrating on testing new features?

Automate functional tests – Selenium

Selenium is an open-source tool to automate web application testing across many platforms. We use it for functional/regression tests. Please read more about Selenium here: http://seleniumhq.org/ . Below are some tips on how to use Selenium in CI.

Figure 4 - Selenium logo

Selenium is divided into two parts:- First, the IDE – an extension for Firefox which allows quick test case development and running of test suites (but only under Firefox) mostly

for debug purpose.

Second, the Remote Control (RC) allows us to run test suites on different browsers. Selenium RC is just an expanded version of the IDE. Below is the functionality that is important in the CI process

- can be run from the command line
- saves reports from test suite execution
- supports most popular browsers

If you want to write test scripts, use Selenium IDE, record your use case, update the script saved by the Selenium IDE, add to the test suite and run it using Selenium RC.

Developing test scripts in Selenium IDE is very simple. Selenium supports most languages, but for beginners or for uncomplicated scenarios, I suggest using HTML. HTML scripts can be converted to any language.

Unfortunately, you can't rely on record and playback feature. Some steps are not saved, or the commands used are wrong. Selenium has a very simple way of adding new commands or extending existing ones. New commands should be added to the user-extensions.js file (writing commands is really simple – believe me). The file should be added to Selenium IDE (*Option->General Tab->* Selenium core extensions field) and to Selenium RC as parameter *user-extension <path to user-extension.js file>*. Many additional commands are written and shared via the Internet by Selenium users. For more information about extensions, please take a look at this example page: http://wiki.openqa.org/display/SEL/Contributed+User-Extensions.

Selenium reports are simple and clear, but unfortunately they need some improvements if we want to use them frequently. Reports show which test cases have failed, and clicking on it moves to the test case, where you can see the status for each step. There are three command execution states: passed, failed and not run. Exploring the report you can see only the command that failed, but what really happened on the tested page we do not know. The failed command is not enough to raise a bug report, so we have to re-run the test (mostly using Selenium IDE). But what if the test passes when we rerun it? What if the test fails on other browsers than Firefox? We would need to re-run the whole suite and observe manually. It's clear that, finding what caused the issue and gathering data/logs takes a lot of time. If we run automated tests often, the aggregate time spent on debugging becomes a critical issue. The solution is to extend Selenium reports. The easier way is to use built in a captureScreenshots function which take automatic screens of tested pages. Before each command is run, the screen of the page is captured and saved. At the end of each command screens are added as a link. When a command fails, it is enough to click on the link to see the page screen and better identify the cause of the error. It is possible to reproduce the whole test case path by clicking on previous commands to see if there wasn't any unexpected behavior not verified in previous steps. Selenium reports can be extended not only by automatic screenshots. It is also possible to save the html code of the tested pages.

delete_all_emails.html

Delete all emails from gmail account		
setTimeout	90000	
open	https://mail.google.com/mail/h/jo4absofnx9z/?logout&hl=en-GB	
waitFotElementPresent	Email	
type	Email	test@gmail.com
type	Passwd	password
clickAndWait	signIn	
selectAllMailsInGmail		
click	//input[@value='Delete']	
waitForTextPresent	conversation	
clickAndWait	link=Sign out	
waitFotElementPresent	Email	
verifyTextPresent	Welcome to Gmail	

Figure 5 - Selenium reports with links to screenshots/saved html code

This part of a report (see figure above of failed test case) shows that Selenium wasn't able to click on the Delete button. But we do not know why. Adding screenshots allows us to check what the state of the page was before the click. A screenshot or saved html code shows us if the page loaded, if there wasn't a Delete button or if the button name changed or maybe other reasons of the failure. Extended reports save time, because we do not need to re-run tests. This is very important if an issue is not 100% reproducible.

Another important thing is to use locators wisely. Selenium allows the use of preferred types of locators, but the default is XPATH. Recording tests usually return XPATH to identify elements like this: //span[2]/center/span/center/div/div/div. It is very difficult for debugging purposes if there are no comments for non-authors as it does not identify the element that should be tested. When testing only a specific element, changes in other parts of the page shouldn't cause failure or influence the test case results.

If a change occurs on the page, but locators are not written wisely, all tests will fail and finding the cause will not be easy. Additionally, all test cases need to be improved not only once for this specific element. It's better to use locator that says something more and contains only the necessary elements. Use XPATH like this: //table[@class='test']//div[text()='Some testing text']. It's simple to read and if the test case fails, finding the cause shouldn't be a problem.

Continuous Integration tool - HUDSON

Hudson is an open-source tool which allows Continuous Integration to be introduced in our process. Hudson uses projects and jobs to manage content. Under each project we can define several jobs. Jobs will be our test suites for a specific project/application. Selenium RC allows us to run the same tests suites on different browsers. So a job will be an individual run for specific browsers.

Hudson supports versioning applications like SVN or CVS. Using them for the management of automated scripts allows us to always run the newest version of tests scripts. Just before each run, Hudson will check if there are any updates, and if any are found, an update is made.

Figure 7 - Hudson logo

Another step is to configure how the job will run. Here are some options:

- jobs are run only by user (on request)
- jobs are scheduled and run on a specific day and hour
- set relationships between jobs

By setting the relationship Build after other projects are built, we can be sure that automated tests will always be run after rebuilding the application. Another way to run tests is to use the schedule feature. Schedule should be used when an application is rebuilt on another server, or when we want to run tests only at a specific time – for example at night.

The Hudson server allows us to manage more than one machine. Agents give us the possibility to use different machines for different projects. It allows us to run tests in parallel or use different environments. For example, if we have one application for testing on different browsers, we have two choices:

- use one server, rebuild the application before each run (job for different browser). This idea, however, has one big minus: the time needed for executing all jobs can be really long when the application and tests are run on the same machine.

- use individual servers for different jobs. In this case we may need several servers (or virtual machines). Running the application and tests on different machines is very important if part of our automated suites are performance tests. Saving time is obvious, but it's important if our test suite is long.

Please check here if you need more info about configuration: http://wiki.hudson-ci.org/display/HUDSON/Use+Hudson.

Integrate automated tests with CI environment

Integration of Selenium with Hudson is quite easy. Mostly it requires us to install the Selenium plug-in for Hudson - http://wiki.hudson-ci.org/display/HUDSON/Seleniumhq+Plugin. You will have to install ant/maven and other applications that will be used to run test suites/rebuild tested application.

For running automated test suites Selenium RC should be used. Ant is used for running tests. As you can see integration is quite easy and does not require any specific knowledge. If we add some new features like screenshots we should write additional Ant targets for them (example – clearing directory, copy Selenium reports/screenshots to archive, ...)

The Selenium plug-in gives two additional pieces of functionality:

- Add link to Selenium report - only for the latest run.
- Add Selenium test result trend chart. Charts gather data from the latest job runs and show trends. We can observe how many test cases failed and how the number of automated tests is growing.

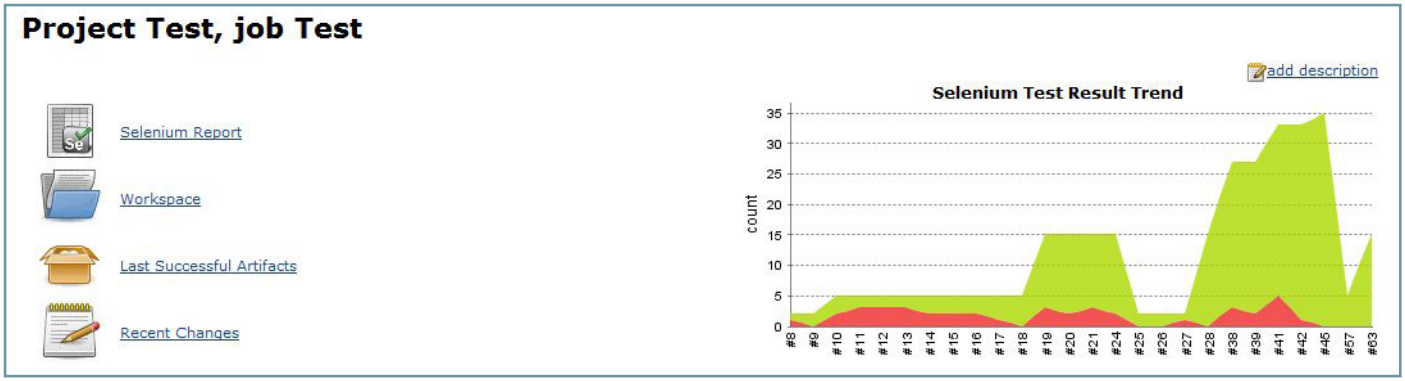


Figure 8: Hudson: Selenium plug-in installed on Hudson, Selenium Report link and Selenium Test Results Trend chart

Time rules

Time is one of the most important subjects when we think about integration of our automated tests with CI. Before we start, try to answer the questions below.

- How often do we want to run/rebuild the application?
- How much time is needed to rebuild applications and run automated tests?
- How much time can we spend on analyzing test results (regression tests)?

The answers to these questions are crucial for us.

How often is it worthwhile to rebuild the application and run automated test cases? Of course, as often as possible. Should it be after each commit? Maybe daily? It depends how many developers are working on a project and on your specific application. Mostly one daily integration should be enough and night-time suites the best. There could be some issues when your application is built by developers from around the world, but the most important factor is the tester's time zone. If we schedule builds at night,

everything should finish before the testers come to work. As a result of this he can check the results, reports bugs and start doing his daily task. If there are any critical issues, Scrum meetings will be a good place to discuss it.

The time needed to rebuild applications and run test suites is very important, too. This time should not exceed 12 hours. If it takes longer, we should divide it into smaller suites or choose important tests which are run daily on week-days with all tests run at the weekend. Getting results everyday and interrupting work to check results can be disturbing for the tester and we should avoid it.

The main test engineer task during each iteration is testing new features. If test engineers spend daily up to 30 minutes on automation (if the automated tests found any issues), it shouldn't interfere with their daily tasks. In total it won't be longer than the time spent on regression at the end of iteration. As a result of this, all critical issues will be found quickly and fixed or raised to the client if needed.

Should we use open-source tools?

Why is it worth using open-source tools? There are several reasons. The example above shows that we can integrate them successfully, we can extend functionality ourselves depending on our needs, or we can just find a solution on web. Considering the money that you might spend on licensing tool products, it may be worthwhile to use it for extending an open-source tool. In most of cases it will be cheaper, and you will have exactly what you need.

Biography

The author of this article has more than 6 years of experience in QA Engineering for big as well as small companies. Currently he works as Head of QA for Cognifide – a digital technology agency.
Zbyszek has been involved in several projects with different sizes and methodology where he could introduce automation. His experience was used to optimizing the time spent on automation getting maximum benefits. This is very important, especially in Agile methodologies, and it's the area in which he specializes.

START THE NEW YEAR WITH A SMILE!

E-Exam or Paper-based*, 4 languages!*
English, German, Spanish, Russian

ISTQB® Certified Tester, Foundation Level
ISTQB® Certified Tester, Advanced Level
ISEB® Intermediate Certificate in Software Testing
iSQI® Certified Professional for Project Management
V-Modell® XT Pro
iSAQB® Certified Professional for Software Architecture
IREB® Certified Professional for Requirements Engineering
iNTACS™ certified ISO/IEC 15504 Provisional Assessor
iNTACS™ certified ISO/IEC 15504 Competent Assessor
TTCN-3® Certificate
ECQA® Certified Innovation Manager
iNTCCM® Certified Professional for Configuration Management
iSECMA® International Certified Professional for IT-Security Management
ISSECO® International Certified Professional for Secure Software Engineering
RFID Management
QAMP® Quality Assurance Management Professional

The first 200 participants who register for any available iSQI exam in January 2011 will receive an early bird discount of 11 %.

Contact our team:

certification@isqi.org
(subject line: "2011")
or via +49 331 231810-24.

*please check availability

www.isqi.org
CERTIFYING PEOPLE

iSQI®
International Software Quality Institute

Scaling up Problems and Solutions for Open-Source Tools

by Mithun Kumar S R

Open-source tools provide a virtual zero investment to organizations with challenging areas to prove the creativity and skills in its customization. However, potential traps in the selection of open-source test tools and during scaling up from pilot projects to stable commercial projects pose a great threat. The pitfalls and possible precautions are briefly discussed in this article.

Cost and ROI

Organizations tend to neglect the total cost of ownership in the scaling-up process. It is a general belief that open source means that the procurement cost is a bare minimum or nil. Estimation suffers a major setback in this assumption.

Administration and maintenance costs will tend to increase dramatically as the project gets increasingly complex. Some end up regretting this miscalculation in the end and come to the conclusion that a commercial proprietary tool could have been a better option.

Features

Most open-source tools, if not all, are designed to serve a specific purpose rather than being generic like the available commercial tools. There is a risk that these features which have been evaluated to be suitable during the pilot run might turn out to be insufficient during scale-up.

The advantage of having the source code means that customization is faster, although dedicated personnel are required to handle such requirements. This might add further to the overheads.

Non-glamorous and unfriendly with the new users

This setback is not apparent. However, people get fascinated in learning something new if the User Interface (UI) is attractive enough. Open-source tools might not have the same charm as commercial tools. Nevertheless, once the users get into the habit of playing with code, they will also realize the adage which goes "Beauty is skin deep".

Training needs for the team

Since most of the features are custom-made, external training support is likely to be very limited. Scaling up would require every team member to know about the tool. Ample resources for training are available through discussion forums, web logs and support documentation. The problem lies in the scattered knowledge source. Compiling the resources together and storing them in a common repository of the organization would also be a challenge.

The user becomes accountable and liable and not the supplier

Almost all suppliers of open-source tools free themselves of the legal hurdle of any liabilities which may occur in using the tool and state this in the End User License Agreement (EULA). Although commercial tools are no different in this aspect, some liabilities do rest upon them to some extent.

This needs to be carefully analyzed before tool selection and scaling, since the liability and accountability now shifts to the user of the tool.

Security concerns

While most support open-source tools because they can raise the security bar according to their needs, it is often neglected that we need equally qualified staff to think from the hackers' viewpoint. In an organization with limited resources, dedicating a person or a team to specialize in security issues could be a setback.

Support

Unpopular open-source programs may receive only a bare minimum of support. With no point of contact for show-stopper problems and bugs encountered in middle of an important execution, the project can end up in disaster.

Tracking intellectual property

With large teams working on the improvement of the tool, care needs to be taken regarding intellectual property and its possible leakage, which might cause harm to the organization.

Difficulty in integrating with stable applications

While commercial tools provide options to integrate with other sets of tools, e.g. for exporting the documentation from a word processor to the tool, the same may be difficult with custom-made applications.

Limited reporting options might also be a setback in the case of test tools, and the limited derivation of metrics could also be a problem to the management.

Although the list of problems may seem frightening, the proper evaluation of the pilot project, with the issues described in this article in mind, would help in selecting the appropriate open-source tool. After all, breaking free from a specific vendor and proprietary software is exciting, isn't it?

Biography

Mithun Kumar S R works with Siemens Information Systems Limited on its Magnetic Resonance Imaging (MRI) scanners. He previously worked with Tech Mahindra Limited for a major Telecom project. An ISTQB certified Test Manager, he regularly coaches certification aspirants. Mithun holds a Bachelors degree in Mechanical Engineering and is currently pursuing Masters in Business Laws from NLSIU (National Law School of India University).

Tellurium Automated Testing Framework

by Haroon Rasheed

The Tellurium Automated Testing Framework for web applications began originally two years ago as a project with a different testing concept from that of the Selenium framework. While most web testing framework software focuses primarily on individual UI elements, Tellurium treats the whole UI element as a widget; calling the element a UI module.

Currently in 0.7.0 release, the core of the project has quickly generated multiple sub-projects as follows.

- **Tellurium UDL**: Tellurium UID Description Language (UDL) Parser.
- **Tellurium Engine**: Based on Selenium Core with UI module, CSS selector, command bundle, and exception hierarchy support.
- **Tellurium Core**: UI module, APIs, DSL, Object to Runtime Locator mapping (OLM), and test support.
- **Tellurium Extensions**: Dojo Javascript widgets and ExtJS Javascript widgets.
- **Tellurium UI Module Plugin (Trump)**: A Firefox plug-in to automatically generate the UI module after users select the UI elements from the web under testing.
- **Tellurium Maven Archetypes**: Maven archetypes to generate skeleton Tellurium JUnit and Tellurium TestNG projects using one Maven command.
- **Tellurium Reference Projects**: Use Tellurium project site as examples to illustrate how to use different features in Tellurium and how to create Tellurium test cases.
- **Tellurium IDE**: A Firefox plug-in that records user actions and generates Tellurium test scripts. This includes UI Module definitions, actions and assertions. The scripts are written in Groovy.
- **TelluriumWorks**: A standalone Java Swing application used to edit and run Tellurium test scripts. An IDE plug-in for IntelliJ IDEA is in development.

Features:

1. **Tellurium UI Module**. Represents the UI being tested. Object uids are used to reference UI elements that are expressive.
2. **UI attributes**. Used to describe the UI instead of fixed locators. The actual locators are generated at runtime. If the attributes are changed, new runtime locators are generated by the Tellurium Framework. Tellurium then self-adapts to UI changes, as necessary.
3. The Santa algorithm. Improves test robustness by locating the whole UI module in a single attempt. A UI module partial match mechanism is used to adapt to attribute changes up to a certain level.
4. The Tellurium UI templates and the Tellurium UID Description Language (UDL). Used to represent dynamic web content.
5. **Groovy Class separation. Example**: UI and corresponding methods are defined in a separate Groovy class which decouples the test code from the Tellurium UI Module.
6. **Tellurium Framework**. Enforces the separation of the UI Module from the test code allowing easy refactoring.
7. **Additional Tellurium Framework Features**:
   - Uses abstract UI objects to encapsulate web UI elements
   - Supports widgets for re-usability
   - Offers a DSL for UI definition, actions, and testing
   - Supports Group locating to locate a collection of UI components in one attempt
   - Includes CSS selector support to improve test speed in IE
   - Improves test speed by Locator caching and command bundling
   - Supports Data-driven test support

Why UI Module?

The UI Module is a collection of user interface elements grouped together. Usually, the UI Module represents the UI object in a format of nested basic UI elements. For example, the Google search UI Module can be expressed as follows.

```
ui.Container(uid: „GoogleSearchModule“, clocator:
[tag: „td“]){
       InputBox(uid: „Input“, clocator: [title:
„Google Search“])
       SubmitButton(uid: „Search“, clocator:
[name: „btnG“, value: „Google Search“])
       SubmitButton(uid: „ImFeelingLucky“, cloca-
tor: [value: „I‘m Feeling Lucky“])
}
```

Tellurium is built on the foundation of the UI module. Tellurium uses UI Module definitions to build Locators for the UI elements at runtime. This makes Tellurium more robust and responsive to changes of internal UI elements. It also makes Tellurium expressive. UI elements can be referred to simply by appending the names along with the path to the specific element. This enables Tellurium's Group Locating feature, making composite objects reusable and addressing dynamic web pages.

The test cases are written in Java, Groovy or plain domain specific language scripts. The UI definition allows the tests to be more expressive and easy to understand within the context of the tests. For example, the tests for the Google Search UI Module defined above would be as follows.

```
type "GoogleSearchModule.Input", "Tellurium test"
click "GoogleSearchModule.Search"
```

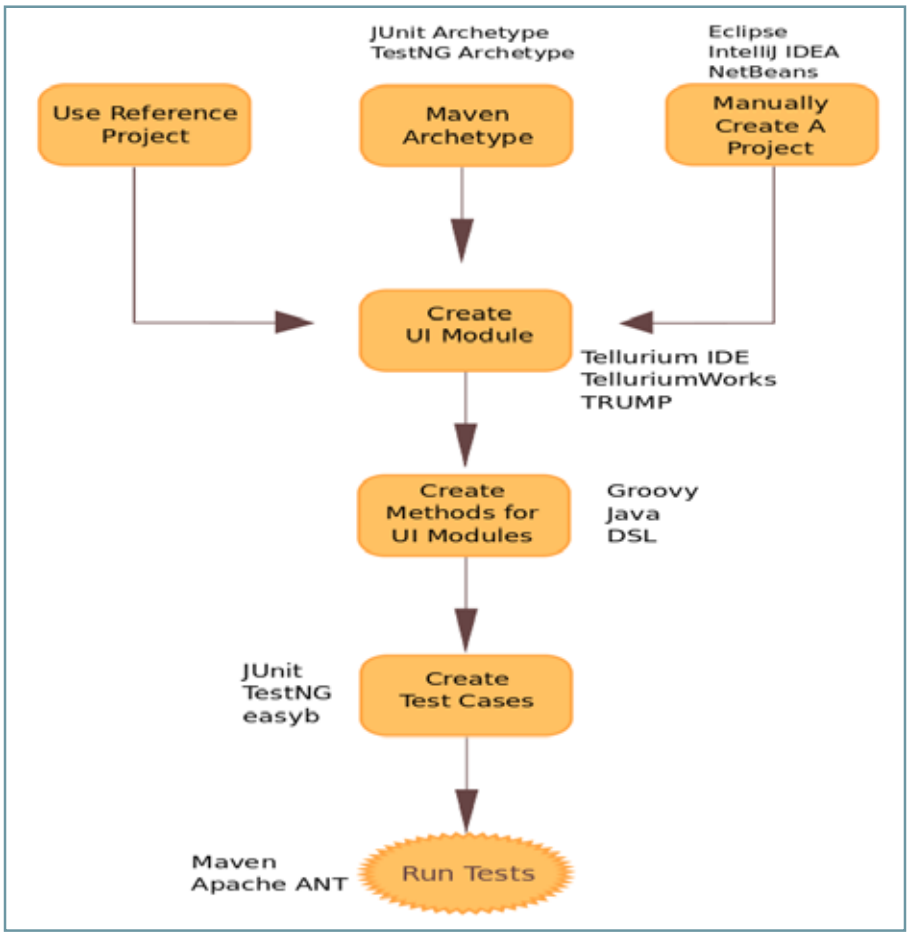
Tellurium sets the object to locator mapping automatically at runtime so that UI objects can be defined simply by their attributes using composite locators.

UI Module definition also decouples the tests from the UI locators, thus making it easier to refactor the tests.

Getting Started With Tellurium

There are several ways to get started with Tellurium. The following diagram illustrates all the possible ways to set up a functional project with Tellurium.

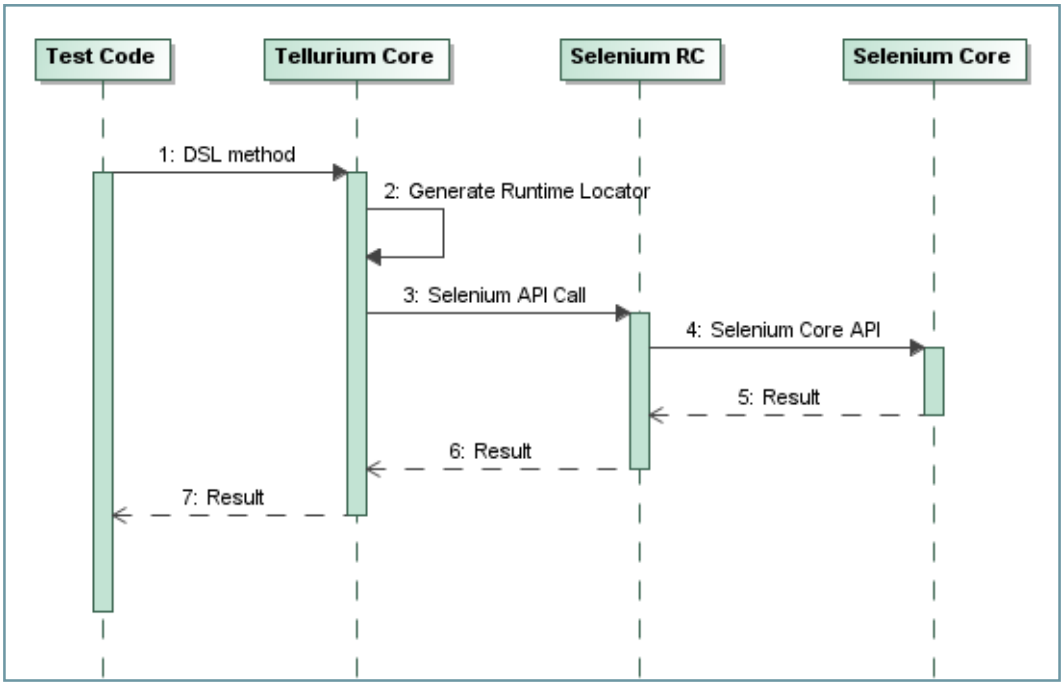
This also serves as the workflow to develop a typical Tellurium test case. A user would start with setting up a Tellurium project, create a UI module using any of the tools provided by Tellurium to identify the UI for the application under test, create re-usable methods for different operations on the UI module and then using these methods in the test case to develop the actual test case.



How Tellurium Works?

Tellurium works in two modes. The first one is to work as a wrapper of the Selenium framework. Tellurium core generates the runtime locator based on the attributes in a UI module and then pass the Selenium calls to Selenium core with Tellurium extensions.

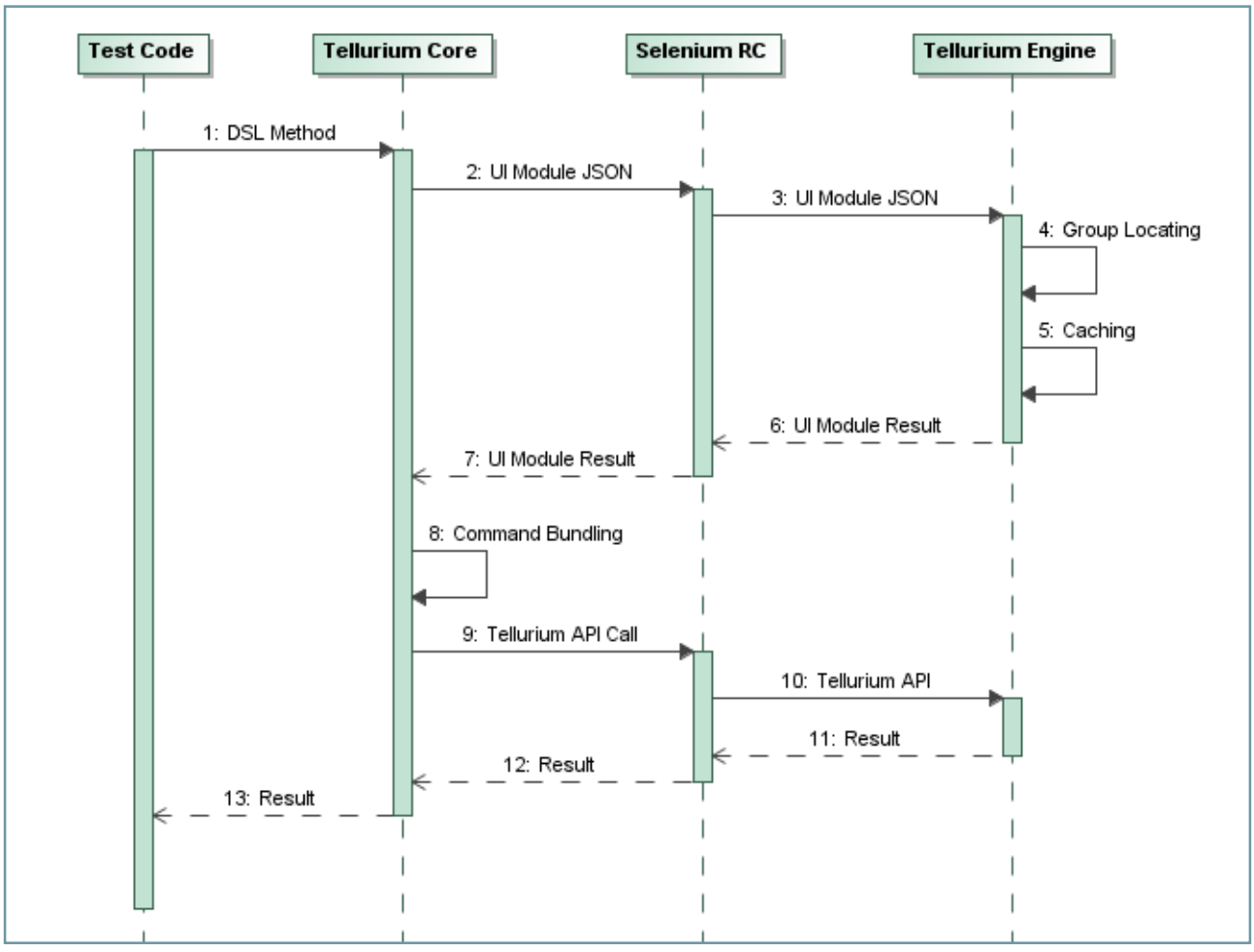
The following diagram illustrates the flow of a Tellurium test when it is run as a wrapper around a Selenium framework.



Tellurium is evolving to its own test driving Engine and the work mode is different as compared to the mode when it is run as a wrapper around Selenium framework.

The Tellurium Core will convert the UI module into a JSON representation and pass it to the Tellurium Engine for the first time when the UI module is used. The Tellurium Engine uses the Santa algorithm to locate the whole UI module and put it into a cache. For the sequent calls, the cached UI module will be used instead of locating them again. Another new feature in Tellurium 0.7.0 is the Macro Command, which combines multiple commands into one batch and then sends them to Tellurium engine in one call to reduce the round trip latency.

This mode is different as shown in the following sequence diagram.



Testing Support

For test support, the following Tellurium Frameworks are used to run Tellurium test cases:

- JUnit . By extending TelluriumJUnitTestCase
- TestNG . By extending TelluriumTestNGTestCase
- easyb . By extending TelluriumEasybTestCase
- TelluriumMockJUnitTestCase and TelluriumMockTestNGTestCase. Incorporated to mock the HTTP server. An HTML web page can then be loaded and tested against it with minimal configuration.

Internationalization Support

Tellurium provides support for internationalization of String and Exception messages. Any software system should have support for regional language settings and options to be effective.

Internationalization and localization provides the support, with locals defining the language and the country. Every locale will have a language code followed by the country. For example ft_FR represents the French language in the country of France.

Internationalized Strings for each locale is provided through a MessageBundle engineered for specific locales and the format is:

```
<MessageBundleName>_<language-code>_<country_code>.properties
```

Future Plans

Tellurium is a new and innovative framework with many novel ideas derived from both the development team and the user community. There are many areas Tellurium is considering for future development, including:

1. Tellurium 0.7.0 has implemented a new test driving engine using **jQuery**. Tellurium will continue to develop the new Engine to its maturity.
2. The **Tellurium IDE** release candidate is out to record and generate test scripts. They are key to the success of Tellurium and they will continue to be improved upon.
3. Tellurium as a cloud **testing tool** is another very important Tellurium future development.

Biography

I have worked as Test Automation Engineer for more than 8 years. During this time, I have worked on different projects ranging from web based CRM applications to Solaris based legacy products for the Oil & Gas industry.
I am currently working as an independent test consultant helping clients to set up maintainable and effective test automation frameworks.

Build your own Open-Source Test Framework

by Robert Bullinger

Some years ago, I was engaged in a case study for a company which wanted to gain first experiences with test case management systems and the management of regression tests by a software tool. Up to then, the company had only used Word documents and simple intranet pages to manage their tests and to document their test results. The costs were to be as low as possible, so only open-source tools were considered in the case study. The schedule was also limited to a maximum of 6 months. In this article, I will describe how we carried out our case study and the benefits we have found to work with a central test management system for regression testing.

1. Capture your needs

Some requirements for the technical environment were given by the company, but not for the functions needed. So we decided to first of all look more closely at the needs of the employees. That's why we arranged short interviews with members of the project management, the test management, some testers and also some developers of the test case automation team.

The most obvious problems in the existing test process where the long time to get reliable results, finding the right test cases for a determined test object and starting them in correct order, and the problems in connection with starting different programs and scripts used to automate test cases with different conditions on different test environments.

2. Evaluate open-source test case management systems

On pages like sourceforge.net and opensourcetesting.org we found the following web-based Systems for our evaluation.

- QaTraq
- RTH
- TCW
- Tesly
- Testitool
- TestLink
- Testmaster
- Testopia

The software was installed on a test server and evaluated in respect of the following areas:

- Usability
- Test case management
- Plan test scenarios
- Test execution
- Test reporting
- Export of test documents
- User management
- Integration of an Issue Management System
- Work performance tests with Use Cases
- Installation
- Software architecture and expandability

Due to some deficits on installation, usability or expandability, not all candidates could be considered. The evaluation for the most suitable software was narrowed down to:

I. QaTraq

II. TestLink

III. RTH

3. Manage regression tests

QaTraq offers a function called test scripts. With this test scripts function you can schedule your test cases for execution. We used this function for our test automation to create test scenarios. Every test case had a field which contained the name of the test automation script; our software read that field for each test case in the test scenario and ensured that the scripts were executed in the right order.

As we had decided that our users should be able to work, manage and configure our new test automation framework completely within the QaTraq GUI, we had to expand it with some new web pages in PHP for the configuration of the test automation scripts, upload it to our version management and then configure the virtual machines and test environments.

4. Create new software components for regression testing

None of our chosen test case management systems had a functionality to execute defined test scripts on determined test environments. We therefore created these components with the .NET Framework in C#. The components which are explained in the following sections were released under GNU GPL 2.0 and are available on request.

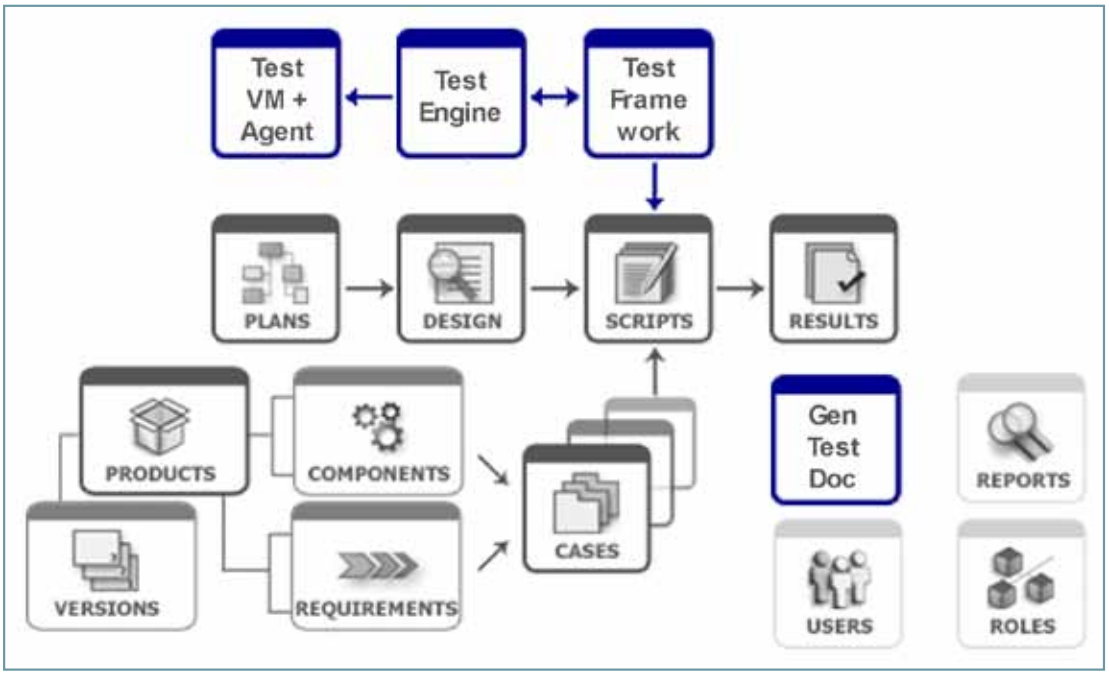


Image 1: QaTraq Test Framework

On our test system we created an agent which offers some services for a central test engine component, like check out or check in some files to a version management system, execute scripts and other software and log events of their execution by the logger tool log4net. The test automation scripts already wrote events into text files and created a report after the tests were finished, so we only had to read these files and transfer their contents and results to our database.

Our test engine for managing the test execution works as the connecting element between QaTraq and our test case execution. It was installed on the same system as our test case management system. Every new execution of a test case or list of test cases was sent to a FIFO stack and the test engine decides on which virtual machine the test could be executed. After that, it calls the service of the test environment agent, checks out all files needed for test execution and starts the test. When the test is finished, it checks in all new files into the version management system and writes the information and results given by the test automation scripts into the database of our test case management system.

To use our test environment capacities more efficiently, and because test cases with longer duration should not block our test execution, the test engine is able to execute different test automation scripts at the same time on different test environments.

5. Reporting

After test execution all the needed information was somewhere in the database, and we wanted to analyze the results. QaTraq already offers some standard templates for creating HTML reports. If these do not meet your requirements, you can expand them with own templates and some SQL-statements. With a commercial upgrade you have much more possibilities and can generate much more detailed reports and export them into images or PDF documents. However, this was no option for us.

We developed another tool which exports the database information to XML and generates a new PDF document with a XSL-FO template. So we were able to generate test documents, status documents and reporting about regression test results and configuration just by one click on a button within our QaTraq extension.

6. Advantages of the open-source test case management system solution

- Central platform for software quality management
- Clear framework for your test process, supported and controlled by software
- Clear responsibilities for test cases, test execution and bug processing
- Unattended regression testing overnight with reliable and repeatable results
- Immediate error reporting during test by e-mail to the persons responsible
- Freeing testers from repetitive work for new tasks, thereby increasing test coverage
- Better and freely adjustable reporting for test manager(s) and project management
- Test documentation can be generated from the database contents in a document template
- Versioned software releases, test cases, test scenarios, test automation scripts, test data, test output, test result logs, test environments. Everything is in full consistency, traceable and managed by the software.
- At best, no work overhead for developers and other users, only the new operating of QaTraq and the framework.
- Existing test automation integrated
- Still expandable for new functionalities
- Reduced costs for our case study and later for regression testing

7. Non open-source implementation

In a later project we used this experience to bring a confused test project back to the route to success. The customer had two departments at different locations with different development processes and tools, both with big quality problems. To report their performance to upper management they collected data about the developed functionalities, created and newly automated test cases and found bugs. However, all the metrics were not set into

relation to each other and made no clear statement about the work progress of the project.

We first developed a uniform test process with quality gates and introduced some better test metrics for test execution and automation. To support the process we used an existing HP Quality Center installation and used its API to integrate a self-developed GUI test automation tool for regression tests. The project management was now able to get a better overview over their project and react faster to unforeseen events. Project milestones were now kept more often. The responsibilities in the test team and the task were clearly distributed, and the test manager could prepare his regression tests by himself, run them overnight and had reliable results the next day.

8. ALM Software

Another way to rape in the benefits of such tools is using an Application Life-cycle management software, e.g. Microsoft's new Visual Studio® 2010.

It offers fully integrated components like requirements management, a complete development and testing environment, test automation, performance testing for web applications, issue management and a project management platform based on a team foundation server with its own central data warehouse.

9. Conclusion

To start working with open-source software is often a very good alternative to commercial software. You can make your first experiences with some tools you are not familiar with and your open- source tool can grow by developing extensions according to your needs. Especially in small as well as in growing teams it could be a big advantage to have such a tool.

For bigger teams open-source could be an alternative, but it may be more economical to buy a tool with the basic functionality you need for your daily work. If at any time you need some special functionality, you could expand your tool by software upgrades or by using some existing APIs provided by the tool vendor and develop these functions yourself.

For large companies a generally used solution should be considered and Application Life-cycle Management software might be a good idea. However, it applies even in this case that companies which have no experience with such software may face some big problems at the beginning, because the functionality could be overwhelming and working with this tool could end in a confusing and money-killing disaster. So take time to collect initial experiences with commercial or open-source test case management tools in small teams and expand step by step to bigger teams, departments and your whole company.

Open-source tools often do not have all the functionalities that commercial tools have, but functionality is not all that counts. It is more important how well the tool supports your daily work and meets your requirements. Therefore both open-source and commercial tools will have to be considered.

Biography

Robert Bullinger is Consultant for Software Quality Engineering at Tesnet Group and member of the Solution Architects. He worked in the automotive, banking, data warehouse and telecommunication sector as project / test manager and test automation engineer. As Solution Architect he is responsible for process management and test process improvement.
He holds a Diploma in software development and several certificates from ISTQB, REQB etc.
On his blog on www.robert-bullinger.de he publishes regularly short German articles about his work and about software quality assurance in general.

Sie suchen das Besondere?

Díaz Hilterscheid

Pros and cons

by Bert Wijgers & Roland van Leusden

Before choosing a tool, it is wise to think about your wishes and needs first, then compare a few candidates and finally, try one or two tools. Even more so if the new tool has to interface with other software. Somehow open source tools are often introduced without any prior tool selection process. Apparently the absence of an initial license fee is all the justification that is needed. However, there are other factors influencing the total cost of ownership of a tool. These factors have to be considered up front, otherwise you may find yourself paying too much and getting too little.

In one of our recent projects it was decided to use an open source bug tracker. This decision was entirely based on the tight budget. The bug tracker was installed on an old desktop computer and all went fine until one day the hard disk of the desktop crashed. There was no backup and the information about our bugs could only be recovered by a hard disk recovery service, with significant costs.

In a tool selection process you have to consider factors that will play a role later on, during the implementation and usage stages. Only when all these factors are taken into account the right choice can be made. In this article we'll focus on tool selection for test automation and performance testing. Since interfacing with the application under test is of the utmost importance in these domains, the tool selection process has to include a trial, often referred to as a proof of concept (POC), in which the tool is put to the test. Only when commercial and open source tools are treated equally in the tool selection process you will be able to choose the one that fits best with your situation.

For every test automation or performance test project it is necessary to identify the testing scope. Is it only for your project or for other projects in the organization as well? How complex is the application under test? Is there just one application under test or are there several? Also, test tools need to fit into the overall testing architecture and they should be viewed as process enablers, not as 'answers'. Based on those requirements the tool comparison can start.

Costs and licenses

Apart from the initial license fees, some other costs of open source software are potentially lower. For instance, open source software typically has lower hardware requirements than commercial alternatives. For example, OpenSTA, used for testing simple ASP pages, is reaching up to 3000 virtual users on a load generator of 1Gb RAM on a single 1Ghz P4 processor with Windows2000. LoadRunner, in the same configuration, can handle a maximum of around 1000 virtual users.

The open source label does not imply the software is free of charge. Developers can and do charge for open source software. Open source actually refers to the availability of the source code. This makes it possible to change the software and adapt it to your specific situation.

Open source software is released under different licenses. One example is the Berkeley Software Distribution license that allows anyone to change the source code, without any obligations. Another example is the General Public License (GPL) that states that any changes you make in the source code must be released to the open source community. If a tool is released under yet another license, be sure to check whether it can be used for commercial purposes.

The Linksys saga is an example of what can happen if you use modified code and don't respect the GPL. In 2003 Linksys released the WRT54G wireless router. Some people from the Linux Kernel Mailing List sniffed around the WRT54G and found that its firmware was based on Linux components.

Because Linux is released under the GPL, the terms of the license obliged Linksys to make the source code of the WRT54G firmware available. It remains unclear whether Linksys was aware of the WRT54G's Linux lineage, and its associated source requirements, at the time they released the router. But ultimately, under pressure from the open source community, Linksys open sourced the WRT54G firmware. With the code in hand, developers learned exactly how to talk to the hardware inside the router and how to code extra features the hardware could support.

If you have plans to modify the source code from an open source tool, in order to implement customized features, you should know under which license the tool was released. Where commercial software requires a license fee, open source software might come with obligations of a different kind.

Wir auch!

Lassen Sie sich anstecken von der kollegialen Arbeitsatmosphäre in einem starken und motivierten Team. Zum nächstmöglichen Termin stellen wir ein:

Senior Consultants
IT Management & Quality Services (m/w) für SAP-Anwendungen
Deutschland und Europa

Sie haben

- eine fundierte Ausbildung oder ein Studium (z. B. Informatik, BWL, Mathematik) sowie mehrjährige Berufserfahrung als IT-Consultant in einem Fachbereich bzw. in der Organisation eines SAP-Anwenders, eines IT-Dienstleisters oder in einem Beratungsunternehmen in entsprechenden Projekten
- ausgewiesene SAP-Kenntnisse (z. B. SAP ERP, SAP BI oder SAP Solution Manager)
- Erfahrung im Customizing und mit ABAP-Programmierung
- Kenntnisse in der praktischen Anwendung von Methoden und Standards, wie CMMI®, SPICE, ITIL®, TPI®, TMMI®, IEEE, ISO 9126
- Erfahrung in der Führung von großen Teams (als Projektleiter, Teilprojektleiter, Testmanager)
- Vertriebserfahrung und Sie erkennen innovative Vertriebsansätze

Sie verfügen über

- Eigeninitiative und repräsentatives Auftreten
- eine hohe Reisebereitschaft
- gute Englischkenntnisse in Wort und Schrift

Dann sprechen Sie uns an – wir freuen uns auf Sie!

Bitte senden Sie Ihre aussagekräftige Online-Bewerbung an unseren Bereich Personal (hr@diazhilterscheid.de). Ihre Fragen im Vorfeld beantworten wir gerne (+49 (0)30 74 76 28 0).

Díaz & Hilterscheid Unternehmensberatung GmbH
Kurfürstendamm 179, D-10707 Berlin
www.diazhilterscheid.com

Mehr Stellenangebote finden Sie auf unserer Webseite:

- Senior Consultants Financial Services (m/w)
- Senior Consultants IT Management & Quality Services (m/w)
- Senior Consultants IT Management & Quality Services (m/w) für Agile Softwareentwicklung und Softwaretest
- Senior Consultants IT Management & Quality Services (m/w) für unsere Kunden aus der Finanzwirtschaft

Díaz Hilterscheid

Platforms and features

Mature commercial software usually offers more features than their open source counterparts. These features range from easier installation and better manuals to script examples and fancy reports. Often they support a wider range of platforms. Currently, open source tools can only test web based applications, while commercial tools can also be used with local client server applications. Ever more often, the front end of these local systems uses internet protocols, like HTTP/S, IMAP and POP3, as well. However, the back end is often a good old mainframe that may still be there in ten years time.

One of the most important features of a tool is the support that comes with it. When a problem arises, your vendor will solve it for you. This may take some time, but at least it's off your mind. With open source tools you have to solve the problems yourself. The community may or may not help you, but anyway, the weight is on your shoulders. When choosing an open source tool for test automation or performance testing you should be prepared to spend more time to get the tool up and running. If you underestimate this extra effort, the open source tool might turn out to be more expensive than its commercial counterpart. On the other hand, if you are able to solve any problem that may arise, why pay for features that you don't need?

Current users admit that open source tooling requires more skill to deploy and maintain, compared to turnkey commercial tooling. But after a steep initial investment in acquiring those skills, the long-term costs are lower. Obviously, cutting out costly licenses saves money for other expenses, like training. However, training for open source tooling is harder to find than training for commercial tools. Only a few open source tools have certification programs

With open source, doing a POC becomes even more important, as open source software is not always documented well and might be harder to install. The level of technical skill required to install the tool might be higher since many open source projects do not focus on installation wizards. Once installed, the more popular open source tools update automatically. With others you have to keep track of the updates yourself.

Testers and developers

With the promise that no scripting is required, commercial tools appeal to testers. Since scripting is their core business, developers are not susceptible to this promise. Developers tend to prefer open source tools to automate their unit testing, since they have the skills and knowledge to adjust the tool to their needs.

In an agile software development environment testers and developers cooperate closer than in a waterfall software development environment. Testers help developers to write their unit tests and developers help testers to automate their functional tests. Testing is considered to be an integral part of the software production process and every team member's responsibility. The role of the tester changes from gatekeeper to team player.

Test automation and continuous integration of new code are essential practices in agile projects. Part of the development cycle might be test first. That is, the tests are written before the code. Developers then write code that passes the tests. This code is subsequently integrated in the product and the tests are added to the regression test suite. The new product is then automatically regression tested. Code base and regression test suite grow together in a highly interactive way.

Agile projects tend to use open source tooling for their test automation purposes. This can be explained from the fact that in agile projects test automation is not the sole responsibility of testers but the joint responsibility of testers and developers working together in multifunctional teams.

Objects and protocols

There is a fundamental difference between test automation tools and performance test tools. Test automation tools work at the Graphical User Interface (GUI) level while performance test tools work at the protocol level.

For a test automation tool, the big question is: "Does it recognize all of the objects in the various application windows properly?" Object recognition for test automation tools is never 100 percent. If object recognition is less than 50 percent, the test automation engineers will be forced to perform so many workarounds that the objective of test automation will not be reached. It will take more time to set up and maintain the test automation tool than it would cost to do the regression tests manually.

For a performance test tool the question is: "Does it recognize the client server communication protocols properly?" This question can only be answered by doing a POC in your own environment, with the application under test. Even if the documentation of a tool, commercial and open source alike, says that the protocol is supported, you might find that it doesn't, for example due to version differences.

Tools and infrastructure

A POC is also useful to compare the test results of the various tools. Test tools are also software, and we know that they contain bugs that can strike at any moment. We have done an experiment with Webload, OpenSTA and LoadRunner installed on the same machine doing the same script in an isolated lab, not connected to the corporate network. Using a single virtual user, differences of 30 percent were found in the reported transaction times. This experiment highlights the fact that the tools don't match with each other. How can we ever know if they match with reality?

The differences in transaction times are caused by the different ways in which the tools work and measure response times. We can get a reliable idea about performance, from either tool, based on the differences in the transaction times under different loads. The absolute transaction times may or may not be realistic. At least, the relative performance under different loads should be realistic. Actual measurements support this. In addition, monitoring tools like Perfmon, Rstatd, Tivoli, HPOpenview, to name a few, will assist you to determine the load you put on your infrastructure.

The tool selection should also take the infrastructure of the application under test into account. If you use a load balancer in your infrastructure, the ability to emulate the behavior of different IP addresses accessing a system (IP spoofing) is very useful. Not every tool supports this. Also, if you have different user groups that access the application through different channels (for example WAN, LAN and ADSL), the ability to emulate the behavior of different network infrastructures is needed. Those kinds of abilities are usually only found in commercial tools.

For final validation you really want to get some time on the production hardware before the application under test goes live, especially when the production hardware differs from the hardware in the test environment. So try to incorporate the production environment in the POC. It should not be too hard to get hold of it when the production hardware is not being used, for example in a weekend.

Along the way, a good performance tester will always question the data that the tools produce. From time to time, he will manually interact with the application under test to see for himself what the user experience is while the application is under load.

A test automation tool or a performance test tool is only a small piece of a much bigger test framework. It may be difficult to integrate an open source tool with the commercial software that you are using. If this kind of integration is needed then it should be part of the POC as well. The same goes for commercial tools. Some commercial tools are vendor locked with other test support tools, for defect tracking, source controlling and test management, thus forcing you to buy those tools as well. You may end up paying more than you thought you would.

Vendors of commercial tools are more than happy to organize a POC to show what their tools are capable of. Also with open source there are parties willing to help you, but their goal is different. They don't want to sell the tool, as it is probably free anyway, but they want to sell consultancy services. If such parties don't exist you should organize a POC yourself. Before you commit yourself, make sure the tool is up to the task at hand and fits in your specific situation.

Preparation and commitment

Before you can start using tools for test automation or performance testing your test documentation needs be at a certain level. Improving test documentation may buy you some time for tool selection and will certainly make your automation project go more smoothly. This step should always be done regardless of the tools you intend to use.

For test automation you might have to state the expected results in the test cases. These are often not documented because an intelligent tester understands what to expect. A test automation tool is not intelligent and has to be told explicitly what to expect. For performance testing you need to know what each user is doing and when they are doing it. In other words, you need to have usage profiles.

Another pitfall is the absence of a dedicated and experienced test engineer. Any which way you choose, it requires time and skill to implement and use tools for test automation and performance testing. If the team can not deliver a dedicated and experienced test engineer the test automation or performance test project is bound to fail, no matter which tool is used.

Foolproof tools don't exist. But if you take some time to think before you start, you can at least choose the tool that is right for you. Be prepared to invest in any new tool.

Biography

Bert Wijgers is a test consultant with Squerist. He has a keen interest in testing methods and integral approaches to software development.

Bert holds a university degree in experimental psychology for which he did research in the areas of human-computer interaction and ergonomic aspects of the workspace. He worked as a teacher and trainer in different settings before he started an international company in web hosting and design for which a web generator was developed. There he got the first taste of testing and came to understand the importance of software quality assurance. Bert has an eye for the social aspects of software development. He uses psychological and business perspectives to complement his testing expertise.

In recent years Bert has worked for Squerist in financial and public organizations as a software tester, coordinator and consultant.

Roland van Leusden is a senior test consultant with Squerist. He is specialized in non-functional testing.

He is passionate about testing and always looking for ways to improve the testing process. He is interested in the people he works with and faces them with a pleasant and targeted attitude. Quality and punctuality are very important to Roland. In close collaboration with the customer he searches for the best test strategy in relation to risk, time and the project specifications. Roland has good communicational skills and the ability come up with creative solutions in stressful circumstances.

Roland has in-depth experience with functional and non-functional testing using various open source and commercial tools.

Does Every Cloud Have a Silver Lining?

Column

by Ian Moyse

To many companies, the sky seems to be the limit with cloud computing. However, they are bound to be disappointed if they fail to see that not all cloud offerings are right for everyone by default. The one-size-fits-all solution doesn't apply, even with something as flexible as cloud computing.

Are you a cloud user?

The idea of cloud computing has become a buzzword, not only within IT departments, but it is also creeping up onto boardroom agendas. Cloud computing itself is indeed clouding many IT decisions today, and it doesn't help when IT managers are presented with many related ideas including SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service). Once these terms are clarified, companies will then be able to move on to choose from services either on a private or public cloud.

Understandably, some customers are put off by the hype of cloud computing, although they might not even be aware that they have been using cloud services themselves – for example, if they have a personal hotmail or gmail account, or if they use social networking sites such as Facebook or LinkedIn.

These are cloud applications that have appeared by osmosis in our lives, without any conscious decision having been made by the user concerning what they are actually using. Their decision is purely based on the benefits they saw, whether that meant mobility of access, or better connection with other people.

Fundamentals for choosing cloud computing

With cloud computing and SaaS expected to enjoy a compound annual growth of between 30 to 40 percent in the next 5 years, vendors are rushing to push their cloud offering. From a customer point of view, it is obviously a good thing to have more choices, although they should be aware that in reality there is a wide range of maturity levels amongst vendors, with some of them just taking their first steps and looking for their first few candidates for experiment purposes.

There are some fundamental elements for organizations to consider before making a decision on whether to deploy a cloud application. First of all, some applications are better suited to be in the cloud than others. For example, email filtering in the cloud is becoming the norm, and it makes total sense to stop all spam and internet malware at the internet level – keeping it at its source. However, it might not be as easy to move a print management software to the cloud if there are 50 printers in the office that need managing.

Secondly, organizations should invest in evaluating the provider they are considering – not only for the brand, but also for their ability and delivery focus on the cloud. A large vendor may seem more attractive, but does it have the pedigree of delivering a service which is very different to the traditional model of selling hardware/software instead of services? Do they have the true capabilities in this area to meet your requirements and deliver a quality service?

If unsure, always ask to see the financials of a potential vendor. This information is useful to find out where the vendor stands in the market, and how healthy it is as a supplier regardless of its size. Also ask about the technology – whether it is owned by the vendor itself or based on someone else's, in which case the vendor might not have 100% control over it.

Things to check before signing up with a cloud service provider:

- What are the terms and conditions in the service level agreement (SLA)?
- Are there penalties if a supplier fails to deliver?
- What has the provider's success rate been over a certain period?
- Can they provide customer testimonials? Can you speak to the customers directly?
- Who is going to support the services? Will it be their own supporting staff or a third party?
- Do they provide out of hours support? If so, what kind of support do you get?
- Where is the data stored? Is it in the UK, Europe, or the USA?
- How often has the vendor updated its service in the past 12 months?
- Will you be getting ongoing value for money from the enhancements?
- Can you see the service roadmap the vendor delivered in the past year?

There is nothing to fear about the cloud if companies perform their diligence as they would when buying any other solution, as long as they know the right questions to ask.

EMEA Channel Director, Webroot. With 25 years in the IT Sector and over 7 of them specializing in security (with 5 in SaaS Security), Ian has been with Webroot over 3 years.

Starting as a Systems Programmer at IBM in the mainframe environment, he has held senior positions in both large and smaller organizations including Senior Vice President for EMEA at CA and Managing Director of several UK companies. Ian has been keynote speaker at many events and has introduced and trained a large number of UK resellers in SaaS Security as well as bringing the technology to many major UK brand name companies. Ian now sits on the Board of Eurocloud and is a contributor to the Cloud Industry Forum (CIF). Ian was awarded the independent AllBusiness Sales AllStar Award in March 2010.

iseb Accredited Training Organisation

1. akkreditiertes Unternehmen im deutschsprachigen Raum

ISEB Intermediate (deutsch)

Der ISEB Intermediate Kurs ist das Bindeglied zwischen dem ISTQB Certified Tester Foundation Level und dem Advanced Level. Er erweitert die Inhalte des Foundation Levels, ohne dass man sich bereits für eine Spezialisierung - Test Management, technisches Testen oder funktionales Testen - entscheiden muss. In drei Tagen werden Reviews, risikobasiertes Testen, Test Management und Testanalyse vertieft; zahlreiche Übungsbeispiele erlauben die direkte Anwendung des Gelernten.

Eine einstündige Prüfung mit ca. 25 szenario-basierten Fragen schließt den Kurs ab. Das „**ISEB Intermediate Certificate in Software Testing**“ erhält man ab 60% korrekter Antworten.

Voraussetzungen

Für die Zulassung zur Prüfung zum “Intermediate Certificate in Software Testing“ muss der Teilnehmer die Prüfung zum Certified Tester Foundation Level (ISEB/ISTQB) bestanden haben UND entweder mindestens 18 Monate Erfahrung im Bereich Software Testing ODER den akkreditierten Trainingskurs “ISEB Intermediate” abgeschlossen haben - vorzugsweise alle drei Anforderungen.

Termine

09.02.11—11.02.11

05.04.11—07.04.11

€1600,00

plus Prüfungsgebühr €200 zzgl. MwSt.

http://training.diazhilterscheid.com

Díaz Hilterscheid

RTH: A Requirements and Test Management Tool

by Bruce Butler, PEng.

Background

When I started working for my current employer as a System Validation Engineer, my first task was to organize the testing documentation and activities for the R&D group. The test documentation that did exist was disorganized: spread across our corporate network and in many different forms, e.g. spreadsheets, Word documents, text files, etc. I needed a tool that would provide a central repository for all test documentation, from test plans to test cases to test results. I have a lot of experience using Borland's CaliberRM to manage requirements on large projects, and at a previous employer I'd found a creative way to use it to also handle test cases and test results, albeit somewhat crudely.

My current employer, however, did not have a budget for such an expensive tool, so naturally I started looking for an open-source solution. I needed to get something up and running within a couple of weeks, and didn't have time to do an exhaustive market survey and try out every available tool. To narrow down my search, I came up with a shortlist of requirements for the tool:

1. Store and manage test cases
2. Store and manage test results
3. Support for multiple projects
4. Support for multiple users
5. Low cost (or free)

I'm a strong believer in requirements-driven testing – I find it a whole lot easier to test something if I know what it's supposed to do. So, in addition to the above list, I wanted a tool that would also let me capture and manage requirements.

Our R&D group already used Bugzilla for defect tracking, so I first looked at Testopia – a test case management extension for Bugzilla. I got our IT guy to install Bugzilla and Testopia on one of my Linux test computers, and I spent a couple of days trying it out. Testopia does a good job of organizing test plans, test cases and test runs and, of course, it integrates very well with Bugzilla. However, I quickly realized that it wasn't going to do what I wanted. Requirements apparently weren't on the radar for Testopia's designers – they appear to have forgotten where those test plans and test cases come from – requirements!

Solution Found!

I next looked at RTH (Requirements and Testing Hub) and quickly realized that my search was over. RTH has a number of features that together provide a comprehensive test management system for multiple projects: a requirements library; a test library; release management; a test results library; and user management. RTH is a SourceForge project, PHP-based, runs on Linux and uses a MySQL database. It provides a web-based interface that is very easy to use. [note: RTH is a test documentation tool – it is neither a testing tool nor a test automation framework]

As I began using RTH on a small embedded software project, I quickly realized that RTH wasn't perfect (I'm a Test Engineer – I can find something wrong with just about any piece of software). There were a number of minor issues (spelling, quirks in the user interface, formatting, etc.) and some missing 'nice to have' features. So, in keeping with the open-source philosophy, I decided to make the changes myself. When I first started digging into RTH's innards, I knew nothing whatsoever about PHP. However, PHP is easy to figure out if you know HTML and have some prior programming experience. It also helped that the PHP code was written in a very organized manner. Once I'd made a number of small fixes, I contacted RTH's Project Manager at SourceForge and sent him my fixes for inclusion in the next release.

How I Use RTH

Rather than describe RTH's many features, I thought I'd describe how I typically use it on a project that has both hardware and software deliverables.

First, I create a new project in RTH. This automatically creates all necessary database tables for the requirements, test cases and test results libraries. Next, I assign the appropriate users to that project so they have access to it.

In our R&D group, we usually write a project's requirements in a Word document and circulate it for review before beginning any design or testing activities. Once the requirements have stabilized, I put them into the RTH Requirements Library. RTH doesn't have a requirements import capability (yet!), so I either enter all the requirements manually using Windows copy/paste or, better yet, find someone else to do it for me. If the requirement is a text-

only requirement, e.g., "The red light shall turn on when switch #1 is pressed", I use a *record-type* requirement. If the requirement can best be expressed in a file, e.g., a mode transition diagram or spreadsheet, I use a *file-type* requirement, which allows importing of any type of file into the RTH database. Each requirement in the Requirements Library has all the basic fields needed for properly managing requirements: name, ID, description, status, priority and several user-configurable parameters. Requirements are version controlled – each time a requirement is modified, its version # gets incremented, and previous requirement versions can be easily accessed. If the original requirements are defined in a hierarchical manner (as many requirements are), I can organize them in the same way, since RTH provides for parent-child relationships between requirements.

With all the requirements in place, I can start populating the Test Library by writing one or more *tests* for each requirement. A test (what I call a 'test case') consists of one or more steps, with each step having an *Action, Test Inputs* and *Expected Results*. Steps can be added, modified or deleted as required using a built-in WYSIWYG editor. Any files I need to support a particular test, e.g. spreadsheets, Word documents, text files, etc. can be uploaded to the test record and are available for retrieval at any time. As with requirements, each test also has a number of fields such as priority, status and several user-configurable parameters. Once I've finished writing all the test steps, I enter an estimated testing *duration* (how long I think the test will take to execute). I then *associate*, or link, that test to the requirement(s) it covers. If the test covers multiple requirements, I can create multiple associations. When making each association, I also specify a coverage amount (0-100%), describing approximately how much of the requirement gets tested by the test.

With the requirements in place and all tests written, I now wait for the development team to give me something to test. Using the Release Management facility, I first create the appropriate *Release* and *Build* entries (e.g. 'Release 1.0', 'Build 123'), based on whatever naming convention the development team uses. Then, for the specific build I will be testing, I define and name a *test set*, where I select which tests from the Test Library I want to run on that build.

Then, it's time for testing! From the *Test Results* page, I select the test I want to run, and RTH displays all its test steps. For each step in the test, I perform the Action and look for the *Expected Results*, then mark the step as Pass or Fail. Once I've performed all the steps, I update the test's status (Pass, Fail, Incomplete, Work In Progress or Awaiting Review). Being a web-based tool, any RTH user with access to the project (such as a nosy Project Manager) can log in, look at the Test Results page and see the current status of all the tests in the test set - providing direct feedback on the testing effort. Clicking on a special hyperlink on the Test Results page will display an estimated time to completion, summing up the duration fields of all un-executed tests in the test set.

If I find a defect (bug) in the product I'm testing, I have to generate a bug report. There are hooks in RTH to interface with Bugzilla, but I haven't gotten around to figuring those out, so for now I have to switch over to Bugzilla to create a bug report (not a big deal).

A Great Way to Communicate

Recently, my employer outsourced some development and testing work to a company on the other side of the planet, and I decided to use RTH to communicate with my counterpart on our sub-contractor's testing team. We gave them restricted access to our corporate intranet, and I created an RTH account for their tester. Once I had imported the project requirements into RTH, the tester's first task was to write tests for all of the requirements. The 13-hour time difference worked to our advantage, as each morning I could log in to RTH and monitor their progress, as their day was just ending. As the tester finished writing each test, he would change its status from 'New' to 'Awaiting Review'. My job was to review all tests marked 'Awaiting Review', add comments (if necessary), then change the status to either 'Rework' or 'Completed'.

When it came to the actual product testing, RTH made it easy for me to monitor the sub-contractor's testing, simply by looking at the pass/fail status of each test on the Test Results page.

Changes Made

Besides making several minor improvements, I've also added two features to RTH. The first is the Requirements Traceability Matrix. Invoking this feature for a particular project generates a report listing all of the tests associated to each requirement, along with the aggregate test coverage for each requirement. Requirements with no associated tests are graphically highlighted, so this feature provides me with a quick way to see if there is incomplete test coverage and whether all the necessary tests have been written.

The other feature I added is the Requirements/Test Report. This report shows, for each requirement in the selected project, the pass/fail status of the most recent execution of all its associated tests. This report is useful in documenting when the testing really is complete. [note: this feature has not been included in the latest RTH release, currently at 1.7.2]

Wish List

The more I use RTH, the more I think of ways to improve it. Unfortunately, my time is quite limited (that darn day job!). Nevertheless, here are a few additions I think would make RTH even more powerful:

- Version control in the test library.
- A requirements 'level' field in every requirement, which captures a simple hierarchical numbering scheme, such as is proposed in IEEE Standard 830.
- A requirements export tool that will generate an HTML, PDF or Word document (RTH currently does export to Excel).
- A requirements import tool.

My long-term goal is to get the requirements library mature enough that it can be used as the sole repository for project requirements – the place where requirements are put when they are first written.

Conclusion

RTH turned out to be a pleasant surprise. I've been using it for just over three years now and have eleven different projects in the database. Not only has it proven useful on active projects, I often use it to retrieve test results from legacy projects.

When writing this article, I focused on RTH's main features only, having skipped many of the details, leaving it to the reader to investigate further. If you think RTH might help with your testing, you can check out its SourceForge home page at: http://sourceforge.net/projects/rth/. You can also find more information, inclu-

ding screenshots and a working version of RTH, at Andreas Schachl's blog: http://requirementsandtestinghub.wordpress.com/. Andreas has been the driving force behind RTH for the past several years.

Biography

Bruce Butler has been working in the Canadian high-tech industry for 27 years in such diverse areas as telecommunications, autonomous underwater vehicles, mining automation, DGPS marine radiobeacon networks, electronic charting and vessel surveillance. He is registered as a Professional Engineer in British Columbia, Canada, is a Senior Member in the American Society for Quality and is certified by the ASQ as a Software Quality Engineer. Bruce holds a Bachelor of Science degree in both Computer Science and Physics from the University of British Columbia and is a developer for the RTH project at SourceForge.

Díaz Hilterscheid

License ISTQB and IREB Training Materials!

Díaz & Hilterscheid creates and shares ISTQB and IREB training material that provides the resources you need to quickly and successfully offer a comprehensive training program preparing students for certification.

Save money, and save time by quickly and easily incorporating our best practices and training experience into your own training.

Our material consists of PowerPoint presentations and exercises in the latest versions available. Our special plus: we offer our material in 4 different languages (English, German, Spanish and French)!

For pricing information, and all other product licensing requests, please contact us either by phone or e-mail.

Phone: +49 (0)30 74 76 28-0 **E-mail:** training@diazhilterscheid.com

Launch of the TMMi® Model

Learn how you can become a TMMi® Assessor

The TMMi® Foundation invites you to participate at EuroSTAR2010
Wednesday 1st December at 8.00 a.m. in Room 17
at Bella Center in Copenhagen

Test Maturity Model integration, TMMi

TMMi maturity levels and process areas

TMMi® FOUNDATION

Join as a member free of charge, and read more about the TMMi® model and the TMMi® Foundation at www.tmmifoundation.org

Participate in the TMMi® discussion on

www.linkedin.com/groups

and search for the

"TMMi® Discussion Group – Official Site"

Participate at the TMMi® Launch at EuroSTAR and listen to:

- Erik van Veenendaal when he introduces TMMi® level 4 and 5
- Fran O'Hara when he unveils how you can become a TMMi® accredited assessor

The TMMi® foundation is a non-profit organization with the main objectives to define and manage the international TMMi® model standard and place it in the public domain

Forgotten tools

by Erik van Veenendaal

In this issue we are discussing open-source tools. Preparing for my column made me almost automatically re-think test tool types, their uptake, my experiences over the years, user satisfaction, benefits achieved etc. Of course, most often we discuss tools that are by far the most popular, such incident management, test management, configuration management and test execution tools. In this column I would like to discuss three types of tools that I have found useful throughout my testing career, but still have a small uptake and are most often not the tools that come to mind first when discussing test tools. I do not intend to provide a full explanation including pro's and con's of these tools, the article is just meant as a reminder to also put them to the front of your mind. I recommend to also consider these tool types when defining a tool strategy, and to not just stick with the more common ones.

Code coverage

> Coverage tool: A tool that provides objective measures of what structural elements, e.g. statements, branches, have been exercised by a test suite. [ISTQB]

Having once been a developer myself, I would have loved to have had such a tool back then. As many others, I thought of some test cases (without much clue as to which parts of the code were executed and which parts not), but if the test cases ran ok, I considered the software that I had written to be ok as well. I believe this way of working is still common in many, if not most, development organizations. My first practical experiences with coverage tooling was in a TV project in 1996 using a non-intrusive freeware

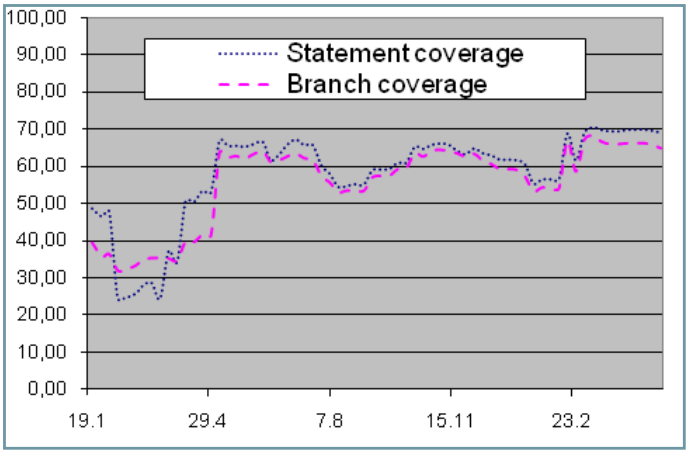


Figure 1: Coverage measurements to guide test improvements

tool. Developers loved it, it supported them in finding out what part of the software had not yet been covered by the tests on a detailed level. I believe most developers are quality-minded like us, but we need to provide them with the knowledge and supporting tools to be able to deliver quality. Coverage tools do exactly that. Of course they can also be used to define strict and measurable exit criteria for component testing. Beware, if you go too strict too soon, otherwise resistance will become an issue. Finally, these tools can also be used for continuous integration when having an automated test suite that runs overnight. We can very easily track the quality of the test suite over time by measuring its coverage (see figure 1.) Nevertheless, recent surveys [PT] show that not even 10% of the development organizations are using a coverage tool. However, with the strong focus on component testing within Agile development methodologies this should change rapidly.

Static Analysis

> Static code analyzer: A tool that carries out static code analysis. The tool checks source code, for certain properties, such as conformance to coding standards, quality metrics or data flow anomalies. [ISTQB]

Many of the arguments that I mentioned when discussing code coverage tools also apply to static analysis tooling. Again, if used in the right way a highly useful tool to support a developer in producing quality software. However, most organization implement static analysis tooling at an organizational level. This may be the preferred situation (according to theory and vendors), but is not always a feasible one. Organizations then end up in endless discussion to get full agreement between developers on style guides, coding standards etc. And what about applying the new style guide and coding standards retrospectively to all the legacy

> Test design tool: A tool that supports the test design activity by generating test inputs from a specification that may be held in a CASE tool repository, e.g. requirements management tool, from specified test conditions held in the tool itself, or from code. [ISTQB]

software that is already in place and will be there for at least the next decade? Not without reason is static analysis in the top 4 for shelfware tooling [PT]. If implementing it in full on an organizational level is asking too much, don't do it!! However, this does not mean that static analysis tools cannot have added value. Perhaps we should keep it much more simple, focus on the twenty or so coding rules we all agree on. Define a minimum set of software metrics with criteria we all agree on such as cyclomatic complexity, number of nested levels and comment frequency, and provide the tool to the developers to start using. I have seen great results in producing more maintainable and more reliable software by applying static analysis tools in a just a limited way. Perhaps the 80/20 also applies here. Remember research has taught us that 40% percent of the failures in the field could have been prevented if static analysis was used. In practice important quality attributes such maintainability and reliability are often forgotten in Agile projects; a static analysis tool that provides support in checking for compliance with the most critical coding rules and software metrics will have added value here as well.

Test design

In many testing courses much attention is given to test design techniques, including exploratory testing. Of course it is important to teach people how to design test cases, to some extent it's the heart of testing. However, recently I read a survey stating that approximately only 50% of the testers explicitly apply test design techniques and around 25% percent apply more than one technique. (Food for thought!) In nearly every test design technique there are steps that would benefit from being, at least partly, automated. Most large test tool providers seem to have no idea what test design techniques are and would benefit largely from an ISTQB Foundation Level course. As a result there is still limited availability regarding test design tools, hence the low uptake. This is a paradox since we perceive it as being a very important part of testing, but tool focus is on test execution, test management and incident management tooling. However, if you start searching you will come across all kinds of easy-to-use tools that support test design techniques. These are not professional tools (there

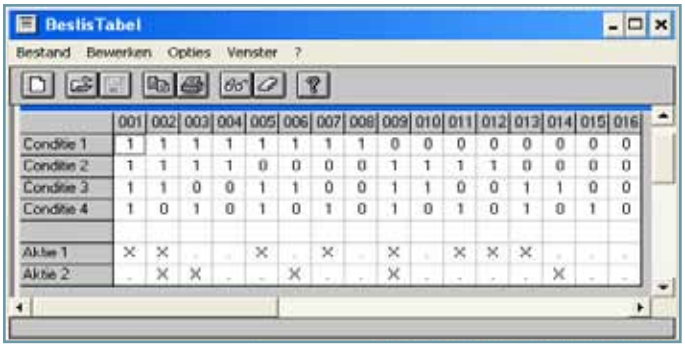


Figure 2: Decision table tool screenshot

are one or two exceptions), but tools developed by someone enthusiastic about testing. In The Netherlands we have a tool called BTWIN which is in fact not more than an advanced Excel sheet, but does support decision table testing (including collapsed tables) perfectly (figure 2). I'm also using a small tool that supports me whenever I have a difficult set of (business) rules that require testing using condition determination coverage; many of the readers are probably familiar with the freeware CTE XL tool that supports classification trees (figure 3), etc. None of these are spectacular tools, but they should be in every tester's workbench as they make the application of test design techniques easier and thus eventually will lead to a larger uptake.

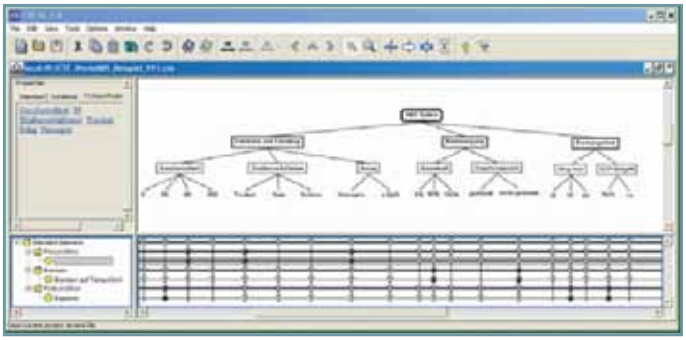


Figure 3: Classification Tree Editor screenshot

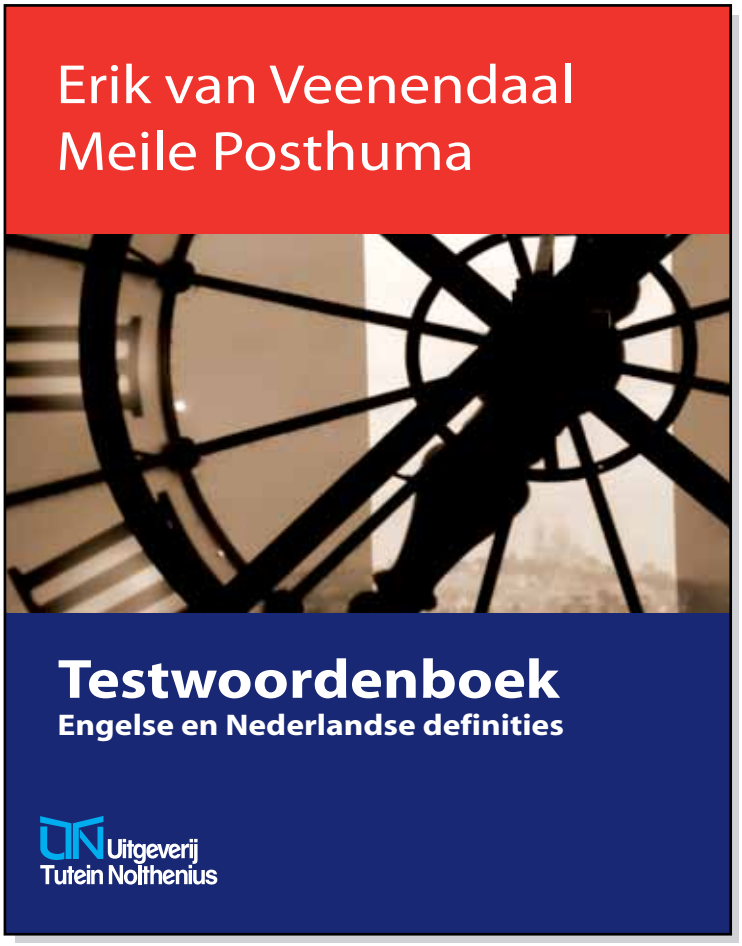
Individuals over Processes

It was only when writing this column it struck me that I was making a case for simple easy-to-use tools over full-blown professional tools. Don't get me wrong, full-blown professional tools offer great support but sometimes there are alternatives depending on the maturity level and domain of the organization, development processes in use etc. In fact in my opinion a test tool strategy can be a combination of both, one doesn't exclude the other. Providing engineers (developers/testers) with a personal tool workbench consisting of easy-to-use and highly supporting tools allows you to get the best out of people. To some extent I'm re-stating "Individuals over processes". Does that sound familiar?

[ISTQB] E. van Veenendaal (ed.) (2010), Standard Glossary of Terms Used in Software Testing Version 2.1, International Software Testing Qualifications Board

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

Erik van Veenendaal (www.erikvanveenendaal) nl) is a leading international consultant and trainer,
and a widely recognized expert in the area of software testing and quality management with over 20 years of practical testing experiences. He is the founder of Improve Quality Services BV (www.improveqs.nl). At EuroStar 1999, 2002 and 2005, he was awarded the best tutorial presentation. In 2007 he received the European Testing Excellence Award for his contribution to the testing profession over the years. He has been working as a test manager and consultant in various domains for more than 20 years. He has written numerous papers and a number of books, including "The Testing Practitioner", "ISTQB Foundations of Software Testing" and "Testing according to TMap". Erik is also a former part-time senior lecturer at the Eindhoven University of Technology, vice-president of the International Software Testing Qualifications Board (2005–2009) and currently vice chair of the TMMi Foundation.



Goede communicatie binnen een project heeft een eenduidig begrippenkader als randvoorwaarde.

Het **Testwoordenboek** biedt:
- een volledige lijst en definities van alle testbegrippen in zowel de Engelse als Nederlandse taal;
- een tabel om snel te kunnen zoeken naar Nederlandse testbegrippen;
- volledige aansluiting bij de ISTQB en internationale testterminologie;
- ter voorbereiding op het ISTQB Foundation en Advanced examen duidelijk geïdentificeerde relevante termen;

Prijs: € 14,90.
Omvang: 148 pagina's.
ISBN: 9789490986018
Te **bestellen**
Bij iedere boekhandel of via **www.utn.nl**

Testen für Entwickler

18.04.11–19.04.11 **Berlin**

Während die Ausbildung der Tester in den letzten Jahren große Fortschritte machte – es gibt mehr als 13.000 zertifizierte Tester alleine in Deutschland – wird die Rolle des Entwicklers beim Softwaretest meist unterschätzt. Dabei ist er beim Komponententest oftmals die treibende Kraft. Aus diesem Grunde ist es wichtig, dass auch der Entwickler Grundkenntnisse im Kernbereichen des Softwaretestens erlangt.

http://training.diazhilterscheid.com

Response Time Accuracy: open-source or Proprietary Performance Test Tool

by Muhammad Dhiauddin Mohamed Suffian, Vivek Kumar & Redzuan Abdullah

Performance testing is a critical test approach in achieving the full coverage of software product testing, apart from functionality and other non-functional aspects. Performance testing will determine that the software under test is capable of performing within an acceptable time according to the end-users' expectations. Our experiences in performance testing, particularly in the load testing of various types of software, ranging from mobile applications, desktop, web-based to cloud-based applications and using different performance testing tools, have triggered us to ask which tool actually gives the most accurate results. The answer is not simple, and it is not that the proprietary tool always gives the right and accurate response times compared to open-source tools, or vice-versa. Thus, before answering the question, it is important to demonstrate that there are differences in response time between various performance testing tools and to understand the tools' behaviors including their influence on the machine resource utilization. We started off by planning the correct test setup for the study.

The test setup involved two identical machines that serve as client with performance testing tools installed and a server which hosts the static web pages. These machines are connected via PC-to-PC LAN cable (commonly referred as cross cable). The reason for connecting it via cross cable is to minimize network factors from influencing the performance test results. The diagram for the setup is as follows:

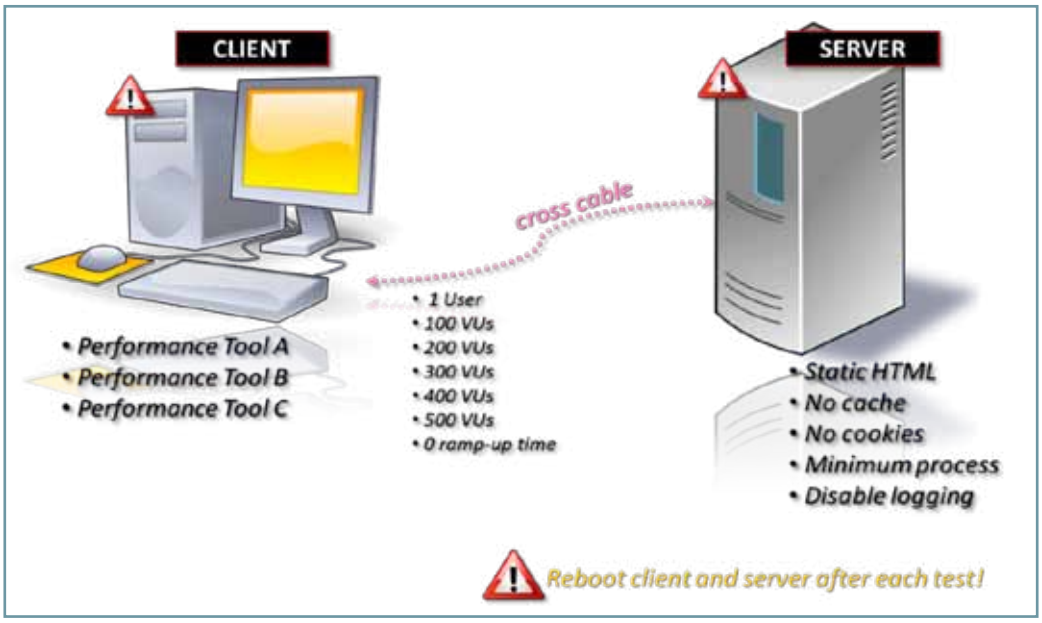


Figure 1: Test setup for comparison of performance testing tools

As shown in the diagram, only three performance testing tools were selected and installed on the client machine. For the scope of the study, the tools are only referred as Tool A, Tool B and Tool C. While the test was running for a particular tool, the services for the other remaining tools were stopped or disabled. As for the server, it only hosted static HTML pages with no cache and cookies. Running processes on this machine were kept to a minimum, i.e. unnecessary processes and server-side logging were disabled. Load testing was performed from 1 user access and increased to 100, 200, 300, 400 and 500 concurrent users with zero ramped-up time. The most important thing for this experiment is to reboot both the client and server machines for each test.

From the technical perspective, the hardware specification for both machines was as follows:

CPU/processor: Intel Pentium D 3.4 GHz

RAM/memory: 2 GB

Hard disk storage: 80 GB

The configuration of the software specification was as follows:

Client machine

Operating system: Windows XP SP2

Java JDK: JDK 1.6.0 update 21

Tool: Tool A (open-source); Tool B (open-source); Tool C (proprietary)

Server machine

Operating system: Windows Server 2003 Enterprise Edition SP1

Java JDK: JDK 1.6.0 update 21

Web server: Internet Information Services 6

HTML page size: 65.8 KB (page: 7 KB; image 1: 25.2 KB; image 2: 33.6 KB)

The test was started with Tool A and with one simulating user accessing the HTML page at the server machine using cross cable. At the same time, we recorded CPU, memory and disk I/O utilization for both client and server machines. The same approach was done three times per virtual user set until the number reached 500 virtual users. The same kinds of load test were performed against Tool B and Tool C. The metrics collected from this exercise were response time, average CPU utilization, memory utilization and disk I/O utilization. The details of the results obtained from this activity are listed below:

a) Tool A

No of Concurrent Users	Response Time (seconds)	Resource Utilization (Client)			Resource Utilization (Server)		
		CPU	Memory	Disk I/O	CPU	Memory	Disk I/O
1	0.103	0%	3%	22%	21%	22%	0%
1	0.100	0%	3%	22%	23%	22%	0%
1	0.103	0%	3%	22%	20%	22%	0%
100	0.258	6%	3%	22%	65%	23%	0%
100	0.29	5%	3%	22%	65%	23%	0%
100	0.231	1%	3%	22%	67%	22%	0%
200	0.198	7%	3%	22%	71%	23%	1%
200	0.231	7%	3%	22%	83%	23%	2%
200	0.538	4%	3%	22%	93%	23%	0%
300	0.337	3%	3%	22%	94%	23%	0%
300	0.265	10%	3%	22%	95%	23%	0%
300	0.229	3%	4%	22%	97%	23%	3%
400	0.169	7%	4%	22%	93%	23%	0%
400	0.534	8%	3%	22%	92%	23%	3%
400	0.278	6%	3%	22%	95%	23%	2%
500	0.394	14%	4%	22%	96%	22%	8%
500	0.35	14%	4%	22%	97%	22%	9%
500	0.306	14%	4%	22%	95%	23%	10%

b) Tool B

No of Concurrent Users	Response Time (seconds)	Resource Utilization (Client)			Resource Utilization (Server)		
		CPU	Memory	Disk I/O	CPU	Memory	Disk I/O
1	0.015	0%	4%	22%	17%	7%	2%
1	0.015	0%	4%	22%	17%	7%	2%
1	0.015	0%	4%	22%	17%	7%	2%
100	1.423	1%	4%	22%	99%	6%	12%
100	1.211	3%	4%	22%	99%	6%	7%
100	1.403	1%	4%	22%	99%	6%	48%
200	3.489	3%	4%	22%	99%	6%	55%
200	4.478	4%	4%	22%	99%	7%	53%
200	5.123	4%	4%	22%	98%	6%	31%

300	6.158	7%	4%	22%	99%	6%	14%
300	7.068	5%	4%	22%	99%	7%	33%
300	5.394	2%	4%	22%	99%	6%	69%
400	8.597	3%	4%	22%	99%	6%	32%
400	8.164	10%	4%	22%	97%	6%	32%
400	8.757	3%	4%	22%	98%	6%	36%
500	11.316	5%	4%	22%	98%	6%	47%
500	11.17	5%	4%	22%	98%	7%	27%
500	8.901	8%	4%	22%	97%	6%	28%

c) Tool C

No of Concurrent Users	Response Time (seconds)	Resource Utilization (Client)			Resource Utilization (Server)		
		CPU	Memory	Disk I/O	CPU	Memory	Disk I/O
1	0.047	0%	3%	22%	35%	19%	3%
1	0.078	0%	3%	22%	38%	19%	3%
1	0.047	0%	3%	22%	25%	20%	3%
100	1.487	3%	3%	22%	100%	19%	59%
100	2.174	3%	3%	22%	100%	19%	14%
100	1.52	1%	3%	22%	100%	19%	22%
200	3.007	3%	3%	22%	100%	19%	27%
200	3.614	2%	3%	22%	100%	18%	15%
200	4.021	5%	3%	22%	100%	19%	38%
300	5.997	2%	3%	22%	100%	18%	22%
300	5.947	3%	3%	22%	100%	17%	10%
300	4.979	3%	3%	22%	100%	17%	18%
400	6.272	11%	3%	22%	100%	22%	15%
400	7.042	2%	3%	22%	100%	17%	6%
400	7.332	3%	3%	22%	100%	17%	27%
500	7.771	4%	4%	22%	88%	18%	7%
500	7.106	7%	4%	22%	100%	18%	24%
500	7.604	7%	4%	22%	100%	16%	10%

Based on the metrics collected from the three tools, we plotted the graphs to compare and understand the trends of the response times. The graphs focus only on comparing the response times under different user loads for three rounds of performance testing. The graphs representing the results detailed in the previous tables are shown below:

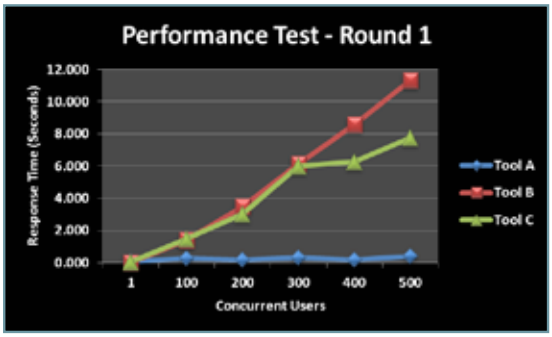


Figure 2: Performance (Load) Test – Round 1

Figure 3: Performance (Load) Test – Round 2

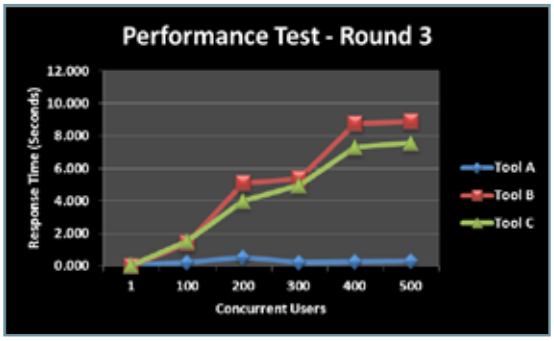


Figure 4: Performance (Load) Test – Round 3

From the graphs plotted, we could observe the differences in response time between the three different tools after running a performance (load) test for the same HTML web page. This result has achieved our research objective, which was to prove that, when using different performance test tools, the response times we get are not similar or even close to each other, especially as the number of concurrent users increases. As mentioned previously, Tool A and Tool B are open-source tools while Tool C is a proprietary tool, which leads us to another research question: "Why do different performance testing tools give different response times when testing the same web page?". In this article, we have demonstrated and proved that differences do exist. In our next research effort we will try to answer why this is the case. Some fundamental understandings that are already in place concern the language with which the tools have been developed, the architecture of the respective tools, the recording mechanism for each tool, capturing and simulating the load used for the performance test, and the method of calculating metrics gathered by each tool. Having this research in place will allow further understanding of the behavior of various performance tools and could help in selecting the most suitable tool for different contexts of testing.

Biography

Dhiauddin is currently a Senior Engineer and Test Team Lead for the test department in one of the leading R&D agencies in Malaysia. He has almost 6 years of experience in the software/system development and software testing / quality assurance fields. With a working experience in automotive, banking and research & development companies, he obtained his technical and management skills from various types of project. He holds an M.Sc. in Real Time Software Engineering from the Centre for Advanced Software Engineering (CASE®), Universiti Teknologi Malaysia. He is a Certified Tester - Foundation Level (CTFL) and Certified Tester Advanced Level – Test Manager (CTAL-TM) of the International Software Testing Qualification Board (ISTQB®). He also has vast knowledge in CMMI®, Test Process and Methodologies as well as Software Development Life Cycle (SDLC). He has just completed his Six Sigma Green Belt project on the test defect prediction model. He has applied and managed various testing strategies in his projects, including functional, performance, security, usability and compatibility testing both for system test and system integration test. His interests are in software engineering and software testing, particularly in performance testing and test management.

Vivek Kumar works as a Senior Test Engineer at one of the most reputable R&D agencies in Malaysia. He has more than 7 years of experience in Software QA both in manual testing and in test automation. During his QA/testing career he has been working in various domains, such as mobile telecommunications, banking, agriculture, semantic technology, and ERP systems. He holds an Engineering degree in Electronics & Instrumentation from the Dr. B. R. Ambedkar University, Agra (Formerly Agra University, Agra). He is a Certified Tester- Foundation Level (CTFL) from the International Software Testing Qualification Board (ISTQB®). He has undertaken and managed various projects from different domains and been involved in the complete project life cycle starting from project kick-off to release to the customer. Vivek has a sound knowledge in CMMI, STLC, SDLC, automation testing and automation framework (keyword driven and data driven) with Selenium, HP Quick Test Professional 10 and Rational Functional Tester. His tireless dedication to the advancement of software testing in general and to performance testing in particular is often referred to as a hobby and not only a job due to the enjoyment he gets from his efforts.

Redzuan is currently the Director of the test department in one of the leading R&D agencies in Malaysia. He has more than 14 years of working experience in software development, technical support and software testing in both local and multi-national organizations.. Redzuan has vast knowledge in various programming languages, CMMI, Six Sigma, telecommunication and project management with international certifications in TOGAF, ASP, Red Hat Certified Engineer (RHCE), British Standard Information Security Auditor (BS7799) and Six Sigma Green Belt among others.

Open Source Automation Tool - Cacique "do it once"

by Rodrigo Guzman

Every Friday the entire QA team spent the whole day to run the same regression test cases over and over again manually. The implementation of each new release depended on the timely completion of these manual tests.

Given this context, if it wanted to reduce time to market and increase test coverage, I could only enlarge the QA team.

We had two options: resign or automate! Luckily no one left the team.

Start with the automation

When we decided to automate, we were faced with several problems.

The tools that existed in the industry were expensive. The tests required considerable knowledge of development to automate and were very difficult to customize and adapt. It was not easy to concatenate multiple scripts.

Most tools can run the tests using the data captured during recording and held within the script. It is difficult to change between different input data using another external source.

After testing several tools, we decided to evaluate the possibility of developing our own framework based on the following requirements:

- Easy to automate testing without knowing how to develop
- Ability to develop and maintain tests (scripts).
- Run tests on different platforms with different input data and external to the script
- Configurable for different work environments
- Easy to concatenate multiple scripts.

"Do it Once" speaks of re-using tests. Our main objective was to create a collaborative tool with a library of scripts and suites to share with other users.

After a year and a half of development, we built the tool "Cacique".

Cacique meets our end-to-end software testing needs. We can scale the test coverage and the reliability of the software, and the tool helps us reduce time to market.

Now we can automate repetitive manual processes of different areas of the company.

We put all our experience in software test automation, creating new features, improving existing ones, making sure that the tool continues to improve, grow and escalate.

Cacique – Open-source tool

We believe that many others may face the same problems of test automation and therefore decided to share the tool "Cacique" with the rest of the community.

Mercadolibre chose to distribute the framework under the GPLv3 license to release the source code of the application and make it freely available for anyone in the community.

Cacique is a web tool that helps you to automate software testing. It integrates a set of tools; Ruby and Selenium are the most important. The base system is developed in RubyOnRails and also has the ability to add „Gems" (libraries language) to add functionality to it, such as libraries to consume resources REST.

The tool also allows integration with other tools, such as continuous integration repositories. We currently use it integrated with Bamboo (Atlassian CI Server).

The biggest advantage is that development knowledge is not necessary to automate testing.

The tool allows the execution of scripts to be easily managed, taking input data from the database in the tool or from other external repositories. It also allows scheduling and can be configured to run on different platforms.

Cacique – Features for non-technical users

Cacique has basic features that help anyone to automate and run tests quickly without requiring development or technical skills

The configuration panel allows the user to choose the interface language to view; the choice is between English, Spanish or Portuguese.

The first step is to record the sequence of the test or task you want to automate. The recording will generate a file in Ruby code. The most widely used tool for this is Selenium (FF plugin). Selenium can record and save the script in .Rb (Ruby).

The second step is to upload the file .rb with the Selenium script into Cacique.

The Cacique interface automatically detects the variables entered by the user during the recording of actions. If the user has entered some data during recording, e.g. name, that variable will be recognized and may be defined as a field variable.

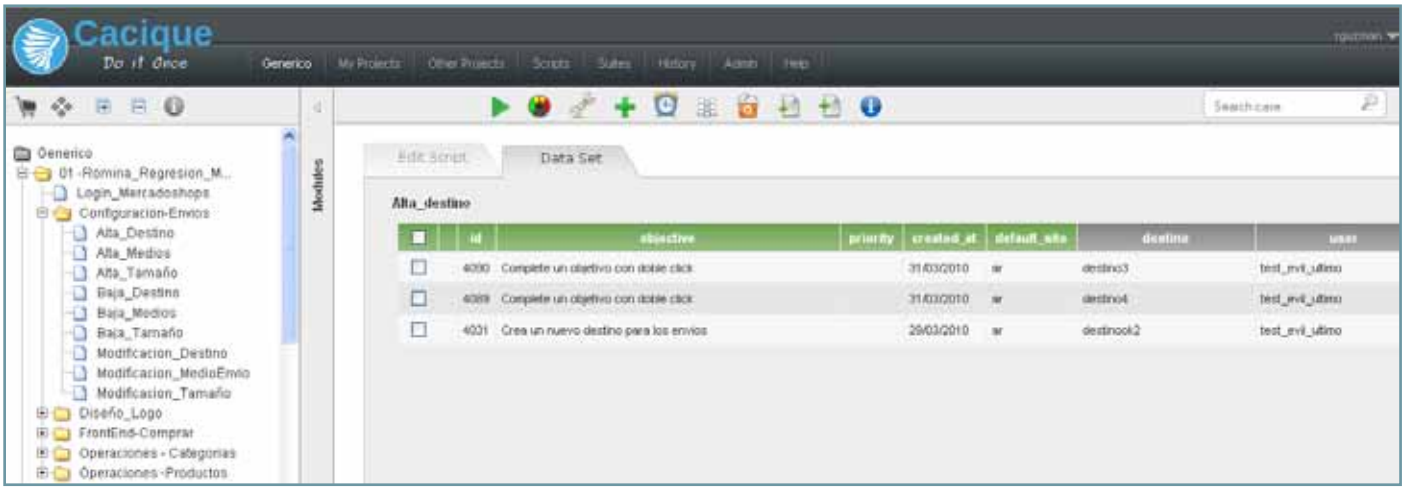
The tool automatically generates a table with all variable fields. These variable fields can then take as many values and variations as the user wants to run.

After this the user can choose the configuration under which to run the automated tests, such as environment, browser, or any other context variable.

The executions can be performed remotely or on the user's own computer.

Finally, all that remains is to run automated tests as often as necessary with different input data needed and then review the results of the runs in the application logs.

While the use of Cacique has many benefits using the library of Selenium, there is no restriction on using any other web-based or desktop library.



(Cacique Screen 1)

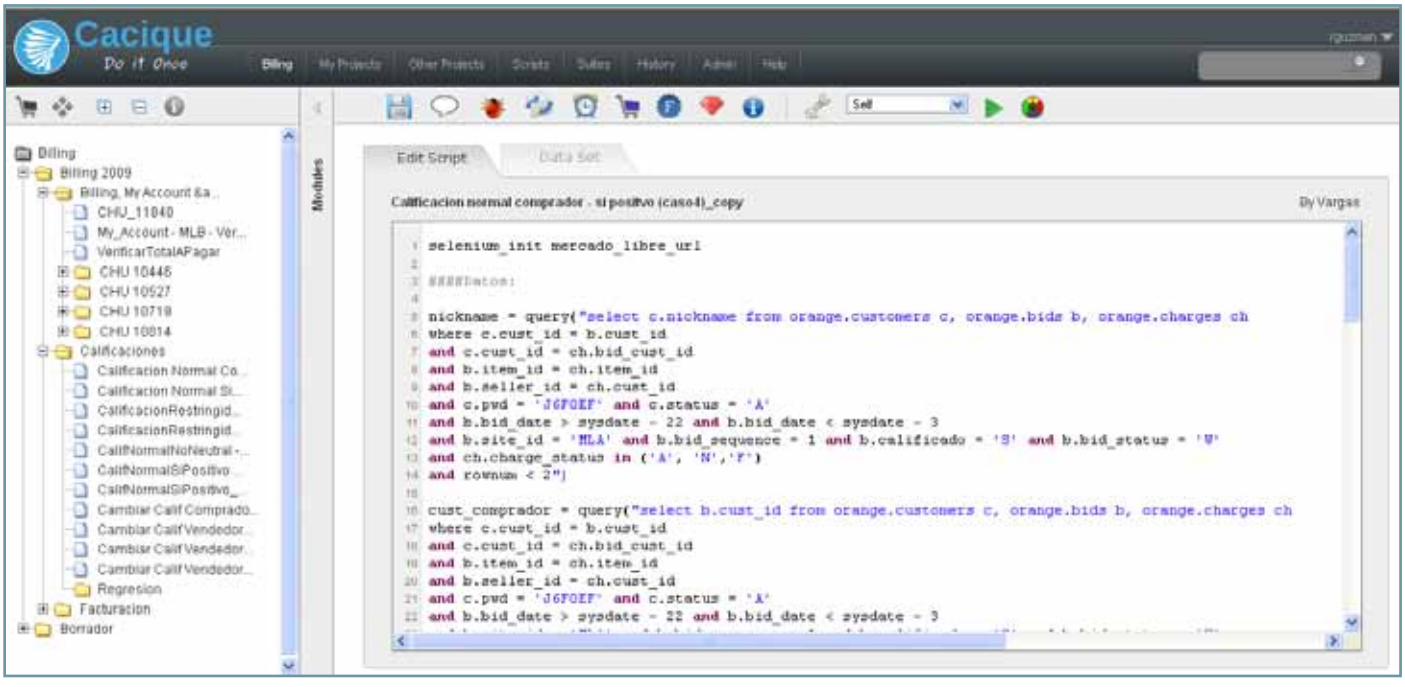
Cacique – Features for technical users

If the user has basic programming skills, he can use the advanced features of Cacique to develop the most robust test automation.

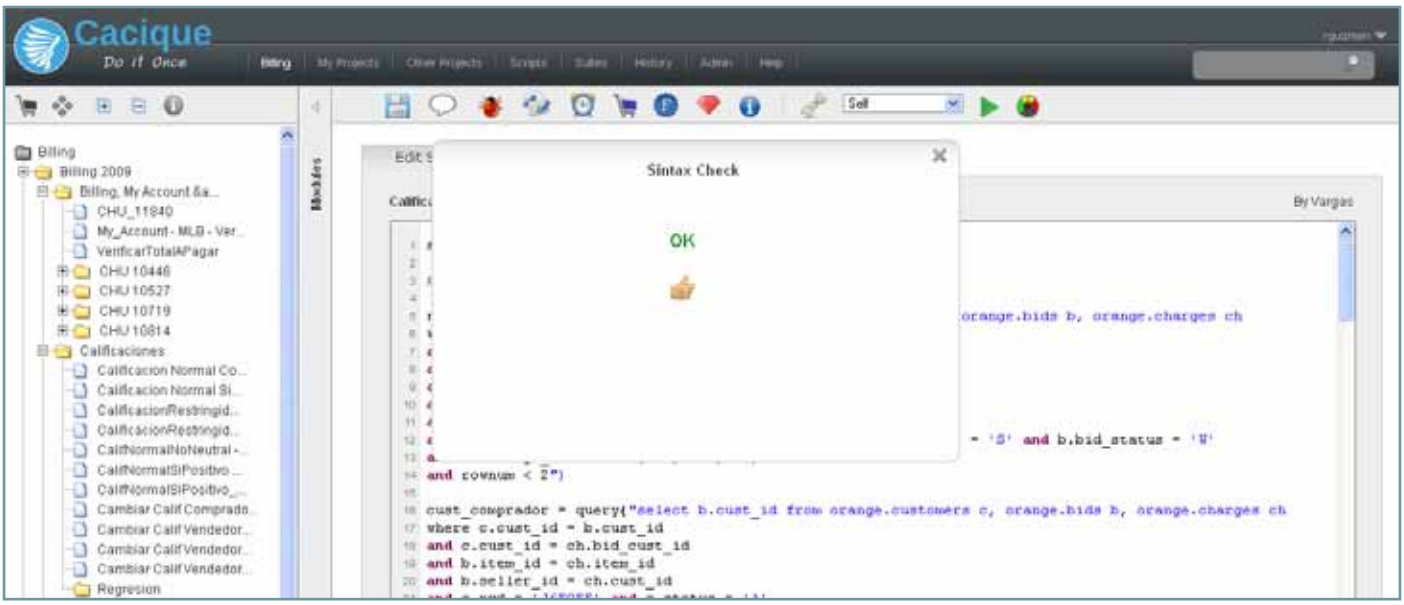
Cacique has a development console which allows the user to include or modify lines of code in the script. It also allows code syntax checking and debugging of the script.

The users can develop an automation script from the beginning, without needing to upload Selenium scripts to Cacique.

Users can also perform functions that can then be shared and reused by other users in their own scripts.



(Cacique Screen 2)



(Cacique Screen 3)

Concatenated test execution

You could set up a test suite.

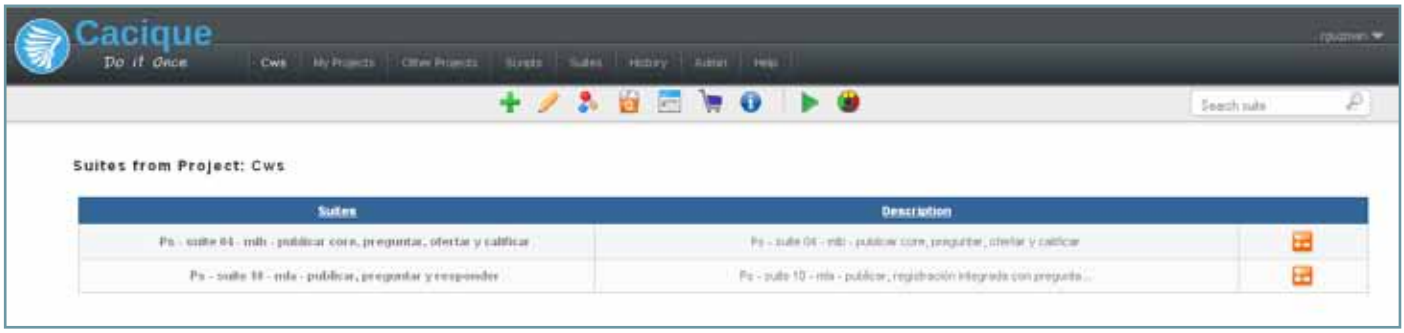
A suite is a set of scripts that run one after the other. The scripts in the suite can be related to share data in the execution. The outputs of a script can be input to the next.

This feature facilitates the execution of automated tests in a sequence.

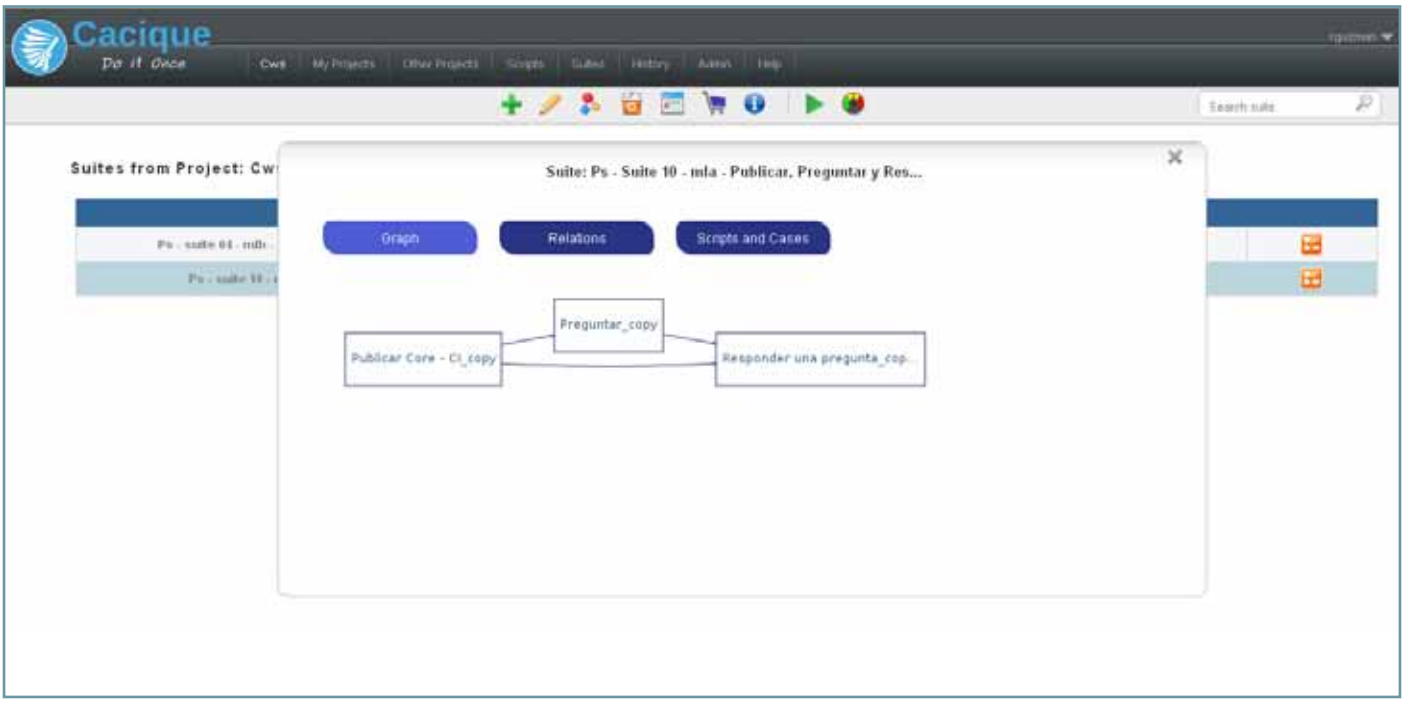
For example, if a user separately automates the testing of the login to an application and then automates the testing of the payment of a transaction, he can then run each script individually or can generate a suite by linking both scripts and the input data, and in doing so he can generate a chain execution.

The suites allow you to easily and graphically link as many scripts as needed and to relate the input data as well.

This feature is extremely useful when running regression testing for end-to-end business test cases.



(Cacique Screen 4)



(Cacique Screen 5)

Run and log

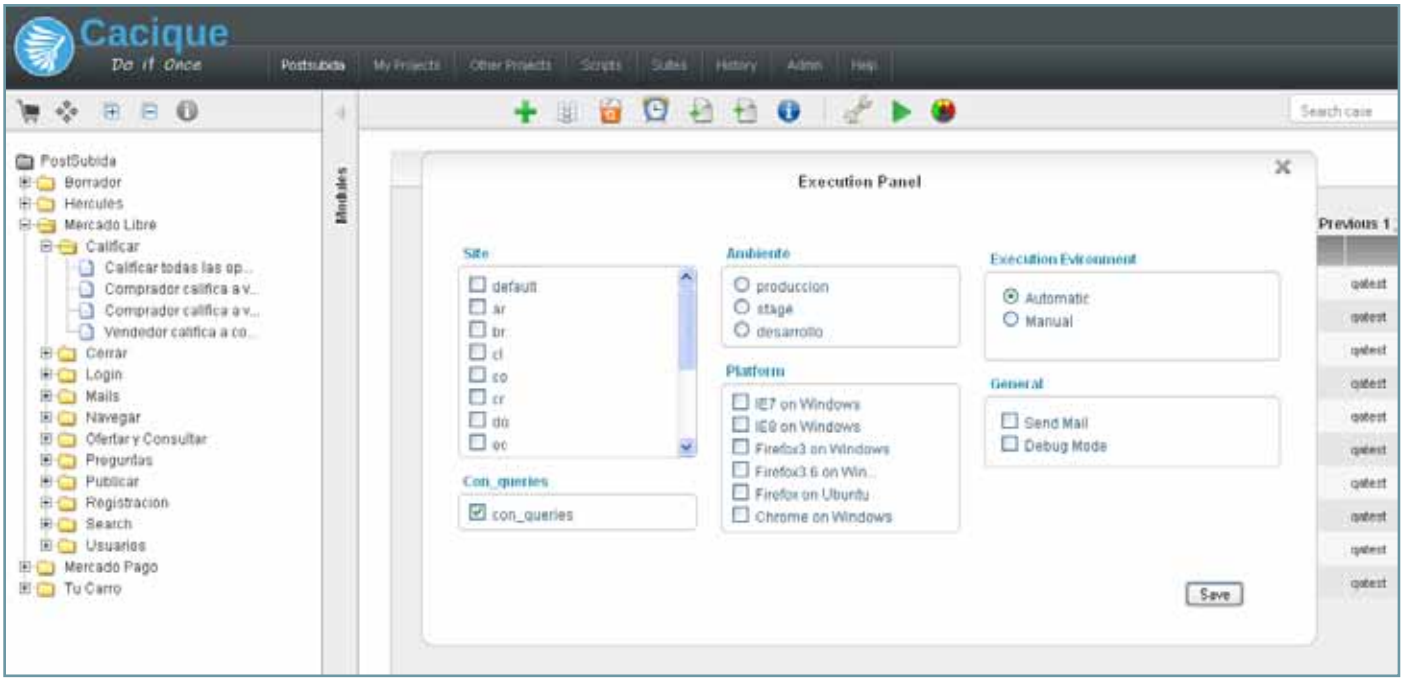
You can enter different data into each of the variables defined in Cacique at the time of loading the script. Each time you want to run a script, you can select one, some or all entries. It is also possible to enter new test data.

You can configure the run via the panel by selecting the context in which you want to run the script and data (for example: working environment, platform, country, etc.).

Clicking on the Run button, Cacique starts the script chosen for each of the selected data in the context set for that run.

After the run, you can check the log output. In the results panel you can clearly see the complete log of each run, case by case.

It can also display the total time it took to run the test. In case of a fault, it can clearly show in which line of code in the script the error occurred and it can display the screenshot that Cacique makes at the time of failure.



(Cacique Screen 6)

(Cacique Screen 7)

Other features

Cacique also includes the following features:

- Program & schedule automatic execution of scripts for specific days and times
- Manage various projects
- View the history of executions
- Define actions to be performed before and after executing a script.
- Connected to different databases as input data and to validate results.
- Can be configured to be called by a continuous-integration server
- Enables communication with other applications through SOAP. (Now integrates with Atlassian JIRA tool)
- Configure the connection of different virtual environments to run the scripts.
- Execute automated tests of APIs.
- Execute automated tests concurrently on different platforms, e.g. Internet Explorer, Firefox, Windows, etc.

Open-source project

Cacique will be available in a free and open-source project from December 2010

Find more information on the website: www.mercadolibre/cacique.com

Also follow on:

Facebook: http://www.facebook.com/#!/home.php?sk=group_127390123983197&ap=1

Twitter: @caciquetest

Cacique

Fernando Crosa, Lucia Brizuela, Horacio Bransiforte, Brenda Guerra, Juan Luna

Currently working in MercadoLibre QA Team.

They are the creators of the tool "Cacique" and the Open Source project developers

All of them have years of experience in software test automation and development under Ruby language.

They apply agile methodologies for the Cacique project management and development of the tool.

Biography

Rodrigo Guzman is Quality Assurance Senior Manager at MercadoLibre.com, a Latin American leading e-commerce technology company. He joined the company in 2004 and is responsible for defining, implementing and managing software quality policies that enable IT to ensure and control the operation of the website in the 12 countries of Latin America.
He holds a degree in Business Administration and a post degree in Quality and Management and has fifteen years of experience in systems, mainly in processes, projects and quality.
Among other things, he implemented the quality management system for the QA department in 2007 and is certified under ISO9001-2000 standards currently in force. He also implemented test automation, project management process improvement and agile practices in IT projects.

Incremental Scenario Testing (IST) – An innovative hybrid of scripted and exploratory testing

by Matthias Ratert

Today's test teams cope with the increasing complexity of the systems under test, while often schedules, budgets and team sizes have not been scaled accordingly. When developing embedded devices such as mobile phones or tablet computers, the functionality has exploded as concurrent applications and multi-modal connectivity over several bearers result in an exponential increase of use cases. Only part of the tests can be automated and, due to limitations in time and budget, only a small fraction of all use cases can be covered in manual testing. Similar observations can be made for automotive infotainment and consumer electronics.

This article describes the implementation of an innovative hybrid of scripted and exploratory testing to support the test teams in managing this complexity and adaptively select and prioritize the tests according to their given budget and previous test results.

The "Incremental Scenario Testing" comes with an open-source tool which guides testers through the test session. After describing the system under test as a simple and generic parametric model, the Incremental Scenario Testing Tool automatically creates high-level test scenarios, which are presented to the tester on the fly while testing. The selection of these test scenarios is continuously adapted with every test session, as the tool learns and recognizes which areas in the test space are critical. Thus it provides the best value for each test session.

Figure 1: The cornerstones of IST

The motivation for developing IST

While developing a complex internet service for a top 5 mobile OEM (supporting a large number of handsets, operators and target market segments, including a multi-backend with e-commerce), we had to specify, execute and maintain hundreds of detailed test cases. With smart test techniques we were able to narrow down the amount of test cases within our database as well as the amount of test cases in our regression test suite. We got a structured test case database to perform regression testing and acceptance testing with the support of an off-shore team and inexperienced testers like interns. As usual, we had a lot of challenges to overcome, including:

1. Frequent changes in requirements caused excess effort in keeping the test specification aligned, as small changes in functionality often affect many test cases.

2. Frequent – and sometimes even ad hoc – software releases didn't give us the time to execute all test cases.

3. Even when we had time to execute all specified test cases, we often missed "important bugs". Despite achieving an impressive pass rate, the quality was still not sufficient.

Our solution was to further optimize the set of test cases and to start with Exploratory Testing.

Our Exploratory Testing experience

The main mission for testers using Exploratory Testing is simply "catching the bugs". By developing charters to explore specific areas we discovered a couple of new bugs, but not as many as expected. It was difficult for the testers to come up with new and creative test ideas on their own, and too many areas remained untested. As the system was very complex, it was often hard to find good charters at all:

- What should be tested / what not?
- What are the critical and risky areas?

- How to ensure good test coverage?
- How to map the recent software changes and their possible side effects to applicable test cases?
- How to implement a convincing feedback loop of the previous test sessions?
- How to cooperate and ensure that some areas are not tested twice?
- How to train testers about system functionality they are not aware of?
- How to motivate even experienced testers to explore new areas?

Getting inspired to create a hybrid

Despite knowing that the testing community has found good answers to almost all our Exploratory Testing questions, we had the idea of implementing the strengths while eliminating the drawbacks of both scripted testing and Exploratory Testing:

- Let's guide and inspire our testers with high-level test stories generated on the fly.
- Let's have a knowledge-based test planning tool, which learns from past test sessions.
- Let's implement intelligent algorithms which automatically choose the most important test scenarios to find as many high severity bugs as early as possible.
- Let's cope with the complexity of modern systems.
- Let's provide visibility of all performed test activities.
- Let's ensure high flexibility for each individual tester and also for the test sessions.
- Let's guarantee platform-independent, web-based tool access.
- Finally, let's keep it simple!

Characterized by high-level scenarios

The obvious challenge was to define a simple but powerful model to provide the test scenarios. First we listed all the functionalities to be tested – in our model we used every possible UI state offered by our system. If a feature has to be tested, every feature state should be listed.

After that we created a list of all possible *Preconditions* which might be dependent or independent of the functionality to be tested. In our case – for a mobile phone – we added several phone settings, the GSM network status, the availability of accessories, and many more.

Finally *Events* were added as an additional parameter to the scenario. The *Event* appends another action to the *Precondition* and *State* and can be independent of the functionality to be tested. In our domain – mobile devices – anything may happen at any time: A phone call comes in, an SMS or MMS is received or the alarm clock goes off. Many bugs can be found by testing these cross-functionalities.

The high-level scenario composed of the three test items *Precondition*, *State* and *Event* are linear sequences of statements in natural language and guide the tester through a test session. The way to reach each specific test item is up to the tester and might vary, depending on the tester's know-how and attitude. This turned out to be very powerful, as some limited randomness is introduced into the test execution, e.g. the input data used, the timing etc.

Adopting our model to the Behavior Driven Development framework "Given-When-Then" or the classic test pattern "Arrange-Act-Assert", the *Preconditions* matches the "Given" / "Arrange". As there might be several *Preconditions* to be fulfilled in parallel, the number of *Preconditions* is kept flexible. The *State* corresponds to the "When" / "Act" and may be linked to requirements or specifications. The *Event* adds another action on top of the "Given, When" / "Arrange, Act".

The "Then" / "Assert" is omitted from the model, as the expected results are not and cannot be specified for millions of possible scenarios. That's also the reason why IST is only applicable for manual test execution and cannot be used for automatic test execution and test result verification.

In Figure 2 some test scenario examples are given for testing mobile originated (MO) and mobile terminated (MT) phone calls. The State is mapped to different MO and MT call states. Everything which might influence and challenge this functionality is added as *Precondition* and *Event*.

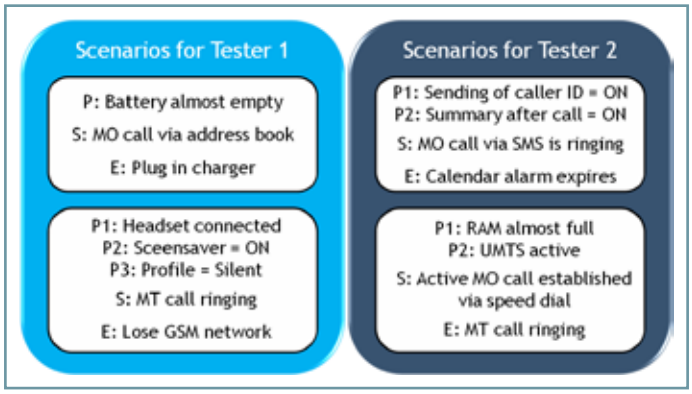


Figure 2: Voice calls scenario examples

As in Exploratory Testing the test results have to be judged by the testers, and if they are not sure whether the observed behavior is expected or at least acceptable, they have to collect the missing information proactively from different sources such as documentation, an experienced tester or developer.

As every combination of one or more *Preconditions*, any *State* and any *Event* is allowed, there will be pairs which are non-reachable or not applicable. To detect and black-list these, the IST tool provides the test result "impossible" next to "passed" and "failed". To obtain an automated and confident acceptance the same scenario is re-tested twice. If all three testers mark the same combination as "impossible", it will never be used again. Otherwise the test manager will get a notification about the conflict for resolution.

The entire scenario description should be understood as a guideline – the tester is free to experiment and to do anything on top of an interesting and inspiring test scenario.

Automating the test scenario generation

Test scenarios are initially generated based on the occurrence probability of each test item. This weight is specified when adding a new test item as 'Often', 'Regularly', 'Sporadic', 'Rarely' or 'Unusual'. We have mapped the weights to the expected likelihood of usage by our customers. The more often a feature is used,

the higher the chance of receiving customer complaints.

As soon as the first execution results are available, the methodology starts learning: Whole scenarios and individual items are prioritized or neglected for upcoming test sessions. Thus, scenarios are evolving over time within the whole project life cycle:

- Scenarios never executed so far are prioritized and the probability of choosing them increases.
- Failed test scenarios are re-tested whereas passed test scenarios are re-tested less frequently.
- Test items being part of failed scenarios are combined with other parameters.
- Failed scenarios and the corresponding items are prioritized based on the reported error severity to reflect the risk.

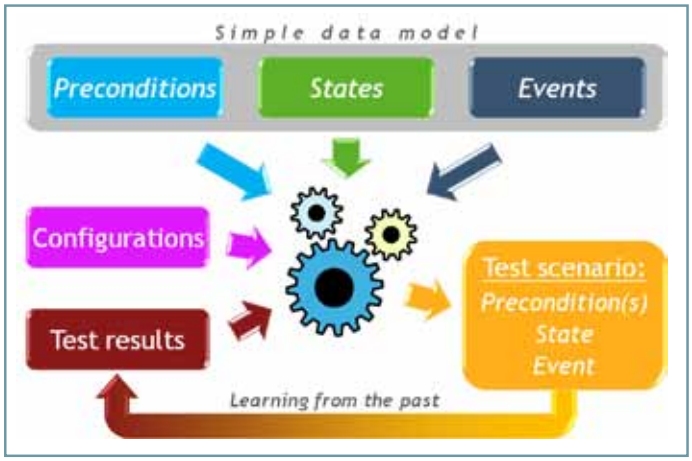


Figure 3: Test scenario generation

After a few test sessions the functionalities (items) never tested as well as the error-prone functionalities are preferred. Using these generation strategies continuously for test scenario executions, the chance of identifying new issues increases significantly independent of the SW development phase.

Involve the developer

Recent changes within the software under test are another important input for generating test scenarios. The developers are best suited to describing the changes in the SW and to identifying possible side effects. However, quite often the link between the development and test teams is missing or is not as good as it should be.

To address this problem, everybody – with developer access to the IST tool – is able to add new items and to prioritize specific items for testing. This functionality has improved our risk analysis to focus testing on recent changes and implementations. As the test subject can be replaced easily and whenever needed, the SW changes are tested immediately after integration.

Managing complexity

Another dimension of complexity is the huge range of different test environments. As we have developed a mobile phone application we had to test on many test setups: Different devices might be used with different versions of the operating system, the localization and operator might change and hardware configurations vary. To assign an individual test environment to each tester, we have defined different *Test Scopes* including several values, e.g. the *Scope* "device" contains a list of all the supported devices. To test with the optimal set of *Scopes*, the tester receives some mandatory *Scopes* together with the testing invitation. The lacking *Scopes* have to be selected by the tester. Within a test session the tester won't change the *Scopes*.

After a while the *Scopes* can be analyzed and prioritized. It is easy to discover the *Scopes that are* not tested or insufficiently supported . As a result testing can be focused without significant effort.

Gaining more flexibility

As the amount of available scenarios is huge, a test session could – in theory – last years. Executing important scenarios at the beginning provides the flexibility to stop a session at any time.

Testers have their own login and can start or continue testing at any time for as long as desired. Every time a tester has some spare time, e.g. whilst the test automation is running or during an hour left before the weekend, he/she might log in and execute some additional scenarios. Or testers are asked to test as much as possible by a certain deadline, e.g. the team has to send the status report by Friday noon. Nobody has to assign tests, nobody has to make sure that everybody has received test charters: The IST system will make sure everybody performs the important tests first.

As the test scenarios and the tool itself are kept simple and can easily be adopted, the teams are scalable and can be distributed. We have collaborated with an off-shore team in Poland, and they enjoyed using the web-based tool in their personal browser.

During and after a test session, several fancy features support the test manager to analyze the work performed. A couple of metrics are provided with different kinds of view. The session results are visualized, individual item and scope statistics are generated and an overall project metric is created.

Error analysis is comfortable as a failed test scenario is re-tested by two other testers within the same test session. By using the same or different Scopes, important information regarding the reproducibility of the found issue is provided. The three individual test results can be easily compared and differences can either be reported via the project-specific defect tracking tool or directly to the developer.

The case study

By utilizing IST in our project we have been able to discover a lot more issues and the customer recognized the improved quality. We have used the test methodology for 1.5 years and have executed around 10,000 test scenarios. That might sound impressive. However, 10,000 out of – in our case – 50 million that are theoretically possible is just 0.02 % of all possible test scenarios. This confirms our aim of getting support in selecting the most suitable scenarios for complex systems, as we cannot execute all of them. Smart algorithms are essential for discovering high-severity bugs as early as possible.

Using IST, even our experienced testers have been motivated to explore new areas and to learn more about our complex system, as some scenarios have been new, even for them. When feeling unconfident either with a scenario description or an expected result, all testers have collected the information proactively.

We have enjoyed the flexible test sessions: Our shortest test cycle lasted less than a day, the longest was almost two months. Usually we started a new test session every week as we received con-

tinuous releases. Within Agile projects the real benefit would be to kick off the test session on a daily basis – the algorithms evolve and testing becomes incremental.

Try it for free

As the "Incremental Scenario Testing Tool" is published as an open-source tool, you can try it for free! For simple projects you might want to use it as your only test management tool. Being part of a complex project it might be the ideal complement for your manual and/or automated regression testing. As the model is kept simple, it can easily be checked whether the system under test can be mapped to it. IST can be utilized early in the project or can be introduced in a later phase.

To start with IST, either download the tool on http://istt.sourceforge.net, or contact the author (Matthias.Ratert@teleca.com) to evaluate it on our homepage. For both alternatives documentation is available to guide you through the setup.

If you like some of the new ideas used within this testing methodology, you are welcome to pick them up, but are requested to add a reference to this methodology.

© iStockphoto.com/ Andresr

Biography

Matthias Ratert is a Project Manager at Teleca in Bochum, Germany. He studied Electrical Engineering at the University of Applied Sciences Münster / Steinfurt. For more than 12 years he has been working in the field of software development, testing and quality engineering for embedded devices. Working for Nokia his responsibilities included the development of software test strategies, the improvement of test automation and the coordination of test managers. He continuously applied and extended this skill set in roles such as Project Manager, Test Expert and Quality Assurance Engineer at Visteon (automotive) and Teleca (mobile communications).

Telling TestStories – A Tool for Tabular and Model-driven System Testing

by Michael Felderer & Philipp Zech

In this article we describe a tabular and model-driven system testing methodology called Telling TestStories (TTS) [1] and its open-source tool implementation.

TTS combines the tabular test definition of approaches like in the Framework for Integrated Tests (FIT) [2] and model-driven testing, i.e., the definition of test models and their transformation to automatic test code. The underlying model is based on a separated system and test model which are interconnected via service calls and checked for validity and coverage. The tool itself has been applied on several case studies so far [3].

We will first discuss the TTS methodology in general and then present a case study to test a Calculator service. Afterwards we will give an overview of the TTS implementation to finally conclude and present future work.

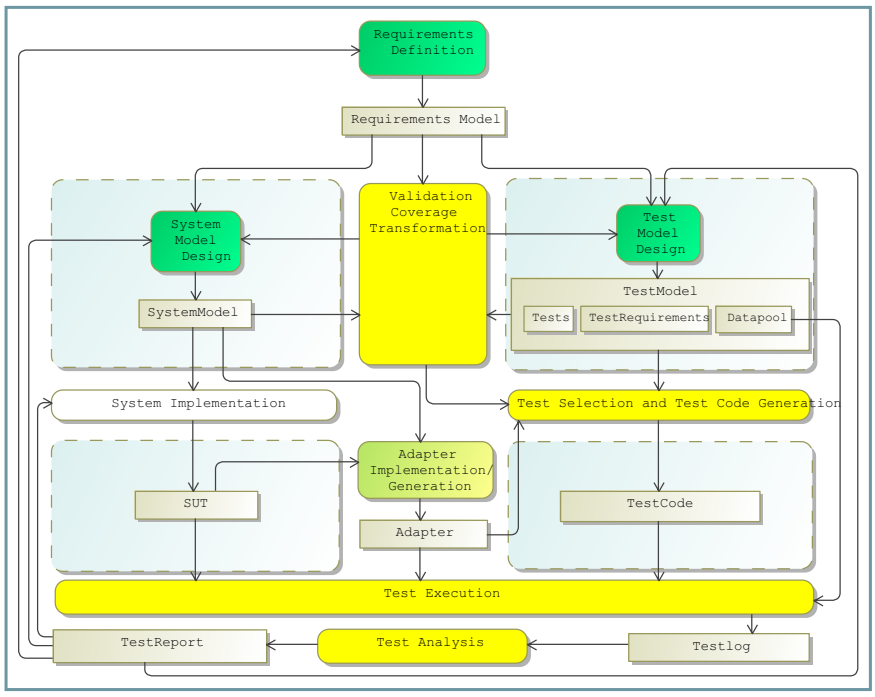


Figure 1: TTS Testing Process

TTS Testing Methodology

The TTS methodology defines a testing process and its underlying artefacts. On the model level TTS defines a *requirements model*, a *system model* containing services as core elements, and a *test model*. The test model contains so-called *test stories* which are controlled sequences of service calls and assertions. In a table assigned to each test story, data for the service calls and the assertions is defined. Each test case corresponds to one line in that table. The notion of a test story is very clear and claimed to be defined by domain experts rather than testers. The models are designed in UML, and checks in OCL are applied to check their consistency, coverage and completeness.

The test stories are automatically transformed to Java code. Its service calls are executable on the running services of the SUT via adapters.

The *TTS testing process* depicted in Figure 1 consists of a *design, validation, execution*, and *evaluation* phase and is processed in an iterative way.

The first step is the definition of requirements. Based on the requirements, the system model containing services and the test model containing test stories are designed. The test model design includes the definition of tests, i.e., test stories or test sequences, the data pool definition and the definition of test requirements, i.e., specific validation, coverage or assertion criteria.

The system model and the test model, the data pool and the test requirements, can be validated for consistency and completeness and checked for coverage. These validity checks allow for an iterative improvement of the system and test quality.

Test stories are normally modelled manually, but in specific cases it is also possible to transform system traces to test stories. The methodology does not consider the system development itself, but is based on traceable services offered by the system un-

der test. As soon as adapters, which may be - depending on the technology - generated (semi-)automatically or implemented manually, are available for the system services, the process of test selection and test code generation can take place.

In the tool implementation, adapters for web services can be generated automatically based on a WSDL description, adapters for RMI access can only be generated semi-automatically. The generated test code is then automatically compiled and executed by a test controller which logs all occurring events into a test log.

Figure 2: Provided Services of the Calculator Service

The test evaluation is done offline by a test analysis tool which generates test reports and annotations to those elements of the system and test model influencing the test result. Additionally the lines of the test table are colored green, yellow or red depending whether the corresponding test passes, is inconclusive or fails.

We have introduced the term *test story* for our way of defining tests as analogy to the Agile term "user story" defining a manageable requirement together with acceptance tests.

The separation of the test behavior and the test data has been influenced by the column fixture of FIT which allows the test designer to focus directly on the domain, because tests are expressed in a very easy-to-write and easy-to-understand tabular form also suited for data-driven testing.

Figure 3: Sample TestStory

Concerning the responsible actors in this development process we can identify the following roles: Domain experts/customers and system analysts cooperate and elaborate the requirements and the tests. System analysts additionally provide the services. The minimal demand to our methodology is that domain experts are able to understand the notation of test stories. Experienced domain experts will be able to define test stories on their own. After the adapters are provided, domain experts and system analysts can repeatedly execute the tests and obtain feedback at their specific level of abstraction.

Case Study

During the development of the Telling TestStories framework, several industrial [3] and non-industrial case studies were performed, among them a *Calculator Service* which is used in this article for demonstration purposes.

The Calculator Service is a simple web services based application, allowing a user to perform basic mathematical operations like addition, subtraction, division and multiplication.

After the creation of a new TTS testing project in our tool, the requirements, the static parts of the system model and the test model are developed. In this demonstration example, the requirements just comprise the correct functionality and interaction of the mathematical operations. As depicted in Figure 2, as a core static part of the system model, the provided services have to be modelled as UML2 interfaces.

The four core functional service interfaces of the Calculator Service (*IAdder, ISubtractor, IDivider, IMultiplier*) provide arithmetical operations, both for *integer* and *float* values. It is shown in the lower part of Figure 2 that TTS also allows the modelling of *pre- and postconditions* for the various operations provided by the Calculator Service (in the concrete example, the pre- and postcondition are both defined over the scope of the `addInt(...)` operation provided by the *IAdder* service).

In more complex examples, the system model additionally contains types modelled in UML class diagrams or workflows modelled in UML state machines or activity diagrams.

Based on the services of the system model, the test model is designed. The test model itself contains mainly two parts. The *test stories* describe concrete test cases, and the *test workflow* defines the order of execution and additional assertion for a sequence of

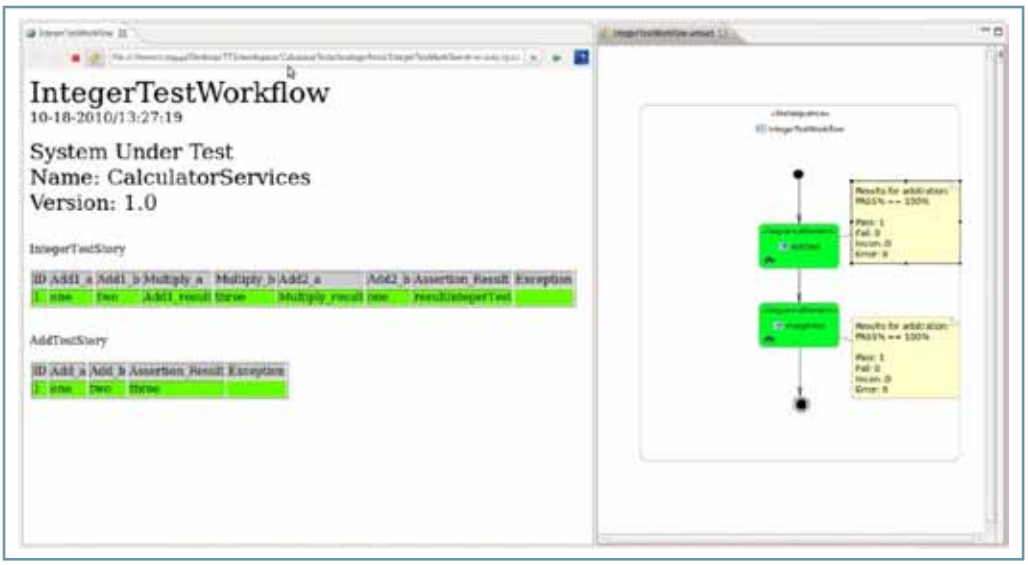


Figure 4: Visualized Test Results

test stories.

Based on functional system requirements, it is then possible to define test stories describing *required* behavior of the SUT. Figure 3 depicts an exemplary test story to be executed on the Calculator Service to test the correctness of the addition and multiplication operations, and their seamless communication.

In a first step the add operation (service call *Add1*) is invoked to add two single integer values. In the next step the result of the addition is passed to the service call *Multiply* for performing an integer multiplication. After a second add operation (service call *Add2*) the final result is evaluated inside an assertion statement. Each test story defined in the test model has an associated table, providing both the needed test data and the final oracle for test evaluation.

The test stories and their corresponding data files are executed as test sequence elements depicted – already annotated with the results of a test run - in the right part of Figure 4.

Test sequence elements define additional assertions that define a global verdict specified on all tests of a test story.

The pre- and postconditions mentioned above define additional constraints that are evaluated during test execution. Being considered a vital component defining the scope of execution of the SUT, they are inevitable for testing, as they allow detailed tracing of a test run in case of failure due to a violated condition.

After the test model quality has been checked and improved by validation and coverage checks with OCL, the test execution phase starts.

Based on automatically generated web services adapters, test code in Java is generated from the test model and executed by the test controller.

Finally, the test results are evaluated offline. In Figure 4 the evaluation result is depicted by coloring test cases in the test data table, by coloring test sequence elements, and by annotating test sequence elements.

TTS Tool Implementation

The case study explained before has been implemented with the TTS tool developed in an industrial cooperation within the Telling TestStories project [3]. Designed as a set of Eclipse plug-ins, the tool consists of various components shown in Figure 5.

The main components correspond to the activities of our testing methodology depicted in Figure 1 and are as follows:

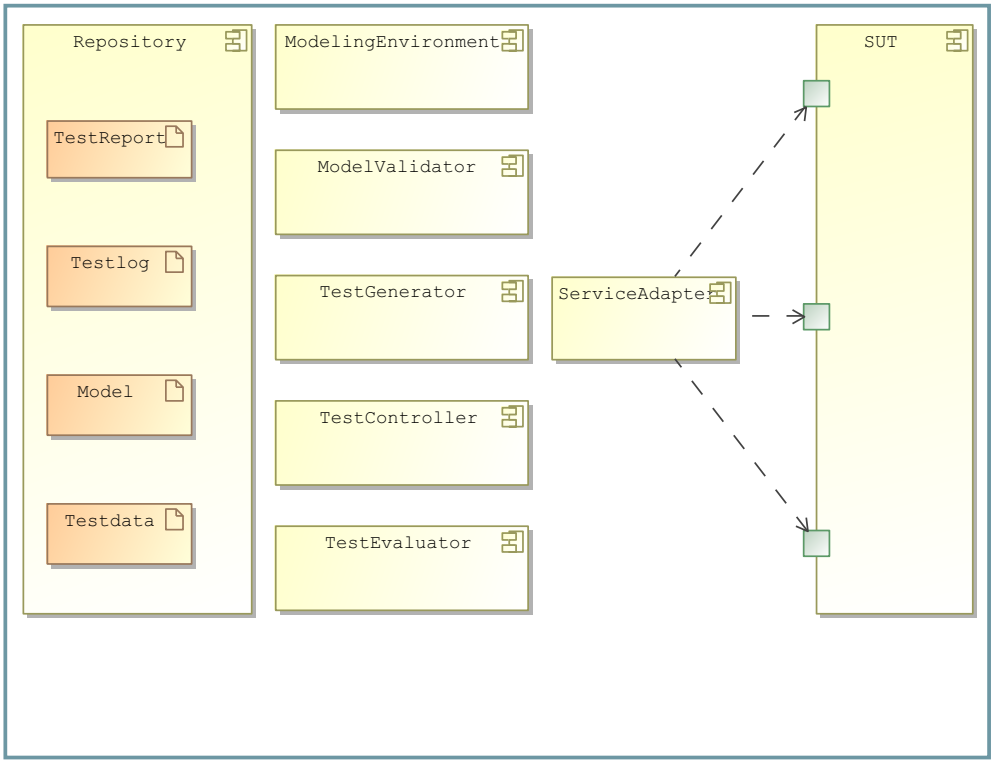


Figure 5: TTS Components

The Modeling Environment is used for designing the UML-based requirements, system model and test model. It processes the workflow activities Requirements Definition, System Model Design, Test Model Design, and Data Pool Definition. The models are stored in the XMI format. There are two implementations of the modelling environment, one based on the Eclipse UML2 Tools editor, the other based on the commercial UML editor Magic Draw.

The Model Evaluator is based on the SQUAM framework (http://squam.info/) and uses OCL as constraint language. It processes the workflow activities Validation, Coverage, Transformation. There is a standard set of consistency, completeness and coverage checks that can be adapted to the needs of a specific project.

The Test Code Generator generates executable Java code and the pre- and postconditions using notions of AspectJ from the test and the system model, respectively. It processes the workflow activity Test Code Generation.

The Service Adapters are used by the test controller to invoke services on the SUT. They can be created manually or generated automatically with tool support depending on the service technology. For web services specified in WSDL, the adapters can be generated automatically. Adapters correspond to the workflow activity Adapter Implementation/Generation.

The Test Controller executes the tests against the SUT. It processes the workflow activity Test Execution.

The Test Evaluator generates test reports and visualizes test results within the test models and the data tables. The test reports are generated with BIRT (http://www.eclipse.org/birt). The Test Evaluator corresponds to the workflow activity Test Analysis.

In its modularity, the tool is only bound to the Eclipse framework. The various components can be exchanged or integrated with other tools, e.g., for test management or requirements engineering.

Conclusions and Future Work

In this article we have presented a tabular and model-driven methodology for system testing. After a general overview of its artefacts and the testing process, we explained the approach by a case study and presented its tool implementation.

The approach is suited for service based systems. Tests are modelled as so-called test stories integrating test behavior and test data.

TTS supports the early definition of high-quality tests by validation and coverage checks. The visual definition of test stories is suited for test definition by domain experts because it is simple and intuitive. After the initial effort of modelling the relevant parts of the system, TTS is claimed to be more efficient with regard to the testing time and the error detection rate.

The TTS tool implementation is distributed under the Eclipse Public License (http://www.eclipse.org/legal/epl-v10.html) and available online at http://teststories.info/.

References

[1] Felderer M., Zech P., Fiedler F., Breu R. (2010) A Tool-based methodology for System Testing of Service-oriented systems. In: The 2nd International Conference on Advances in System Testing and Validation Lifecycle (VALID 2010)

[2] Mugridge R, Cunningham W. (2005) Fit for Developing Software: Framework for Integrated Tests, Prentice Hall

[3] Felderer M., Fiedler F., Zech P., Breu R. (2009) Model-driven System Testing of a Telephony Connector with Telling Test Stories. In: 12th International Conference on Quality Engineering in Software Technology (CONQUEST 2009).

Biography

Michael Felderer is a research assistant at the Institute of Computer Science at the University of Innsbruck, Austria. He is a member of the Quality Engineering research group, he holds a master degree in computer science and is currently finishing his PhD on model-driven system testing. His research interests are model-driven testing, risk-based testing, security testing, model engineering, and software quality in general. Michael Felderer leads the research projects Telling TestStories and MATE to define a model-driven testing approach for service-centric systems, and contributes to other research projects in software testing. He is author of several research papers and speaker at software testing conferences. Besides his research activities, he qualified as a Quality Assurance Management Professional (QAMP) in 2009 and provides test consulting.

Philipp Zech is a research assistant at the Institute of Computer Science at the University of Innsbruck, Austria. He is a member of the Quality Engineering research group, he holds a master degree in computer science, and recently started his PhD. His current research focus lies in the area of risk-based security testing of service-centric systems and cloud testing. Since its initial launch in 2008, he has been a core member and main contributor of the Telling TestStories project.

Berlin, Germany

IT Law
Contract Law

German
English
Spanish
French

www.kanzlei-hilterscheid.de
info@kanzlei-hilterscheid.de

k a n z l e i h i l t e r s c h e i d

Take a daily photo of your development process!

Improve your development process with a Continuous Integration System

by Antonio Robres Turón

One of the major problems in the software development process among the development teams is the integration phase. In this phase the different software components developed by different persons must be integrated, and several problems are often detected.

There are several solutions to solve these problems. The most common solution is performing the integration in phases (within the software development process), another solution is starting the integration at the beginning of software development.

Now imagine you can obtain a built and integrated product every day. Imagine an automatic system that compiles and integrates the source code every day. Many of the integration problems would be solved quickly!

One possible solution is implementing a Continuous Integration System. These systems take the code from the CVS (Concurrent Version System) or repositories. They compile all the source code and finally integrate all the components of the project. There are many applications to implements this system; many of them are Open Source applications, such as Hudson.

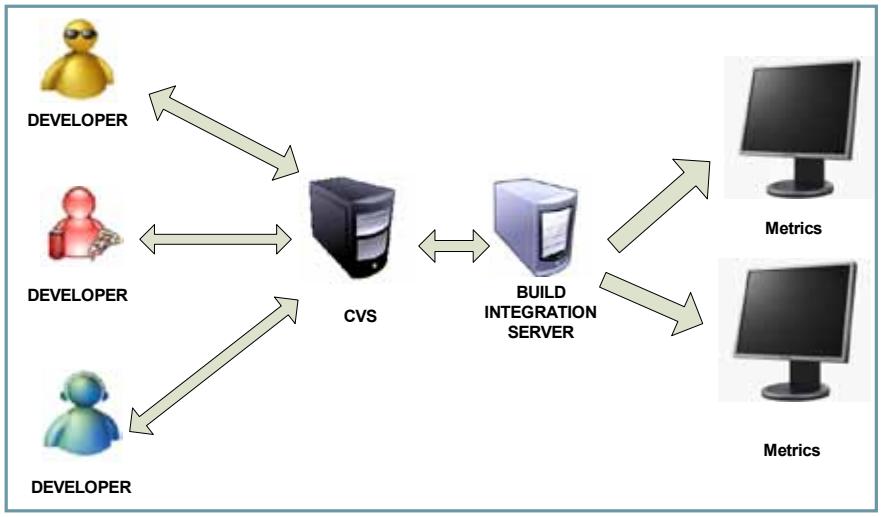
In this article we will share an overview of open source Continuous Integration System called Hudson. Moreover, we will obtain daily metrics to help to improve the development process and the quality of the product.

How it works?

The Continuous Integration System has a simple and intuitive process divided in four phases:

1. The Continuous Integration server connects with the repository and downloads the source code created by the development team and the unit tests.
2. All the source code of the project is compiled and integrated between them by the dependencies rules. The result is an integrated, built product.
3. The unit and integration tests are executed in the Continuous Integration server.
4. Quality analysis tools are executed to obtain the metrics.

The architecture of the system is shown in the diagram below.



All the process can be automatic and scheduled to perform a periodic build (normally daily). Moreover, it can be executed manually to obtain an integrated version at any time. The process is easily configurable and flexible. It allows at any moment the selection of the components in order to compile or integrate and executes the unit and integrated tests.

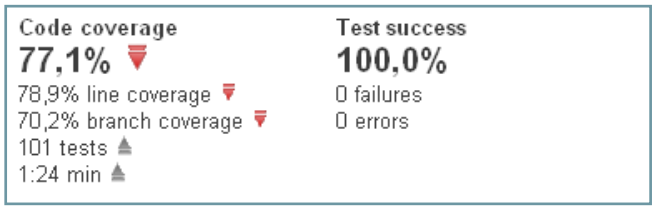
Hudson provides a useful interface showing the result of every compilation and execution done with the historic results. Therefore it facilitates the analysis of every module build. The next picture shows the principal view of Hudson showing the different components, the state of the last execution (red, green, yellow or blue), the stability of the last executions (weather icon), the last correct build, the last fail and the duration of the last execution.

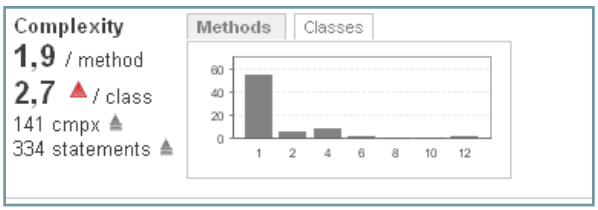
Now you can obtain every day a functional and integrated software to deliver to the testing team. However, we can also extend the Hudson capacities with quality plug-ins.

Obtain a daily quality metrics

One of the advantages of Hudson is the flexibility to load different plug-ins to execute several tools. Many of them are able to obtain quality metrics depending on the code language used.

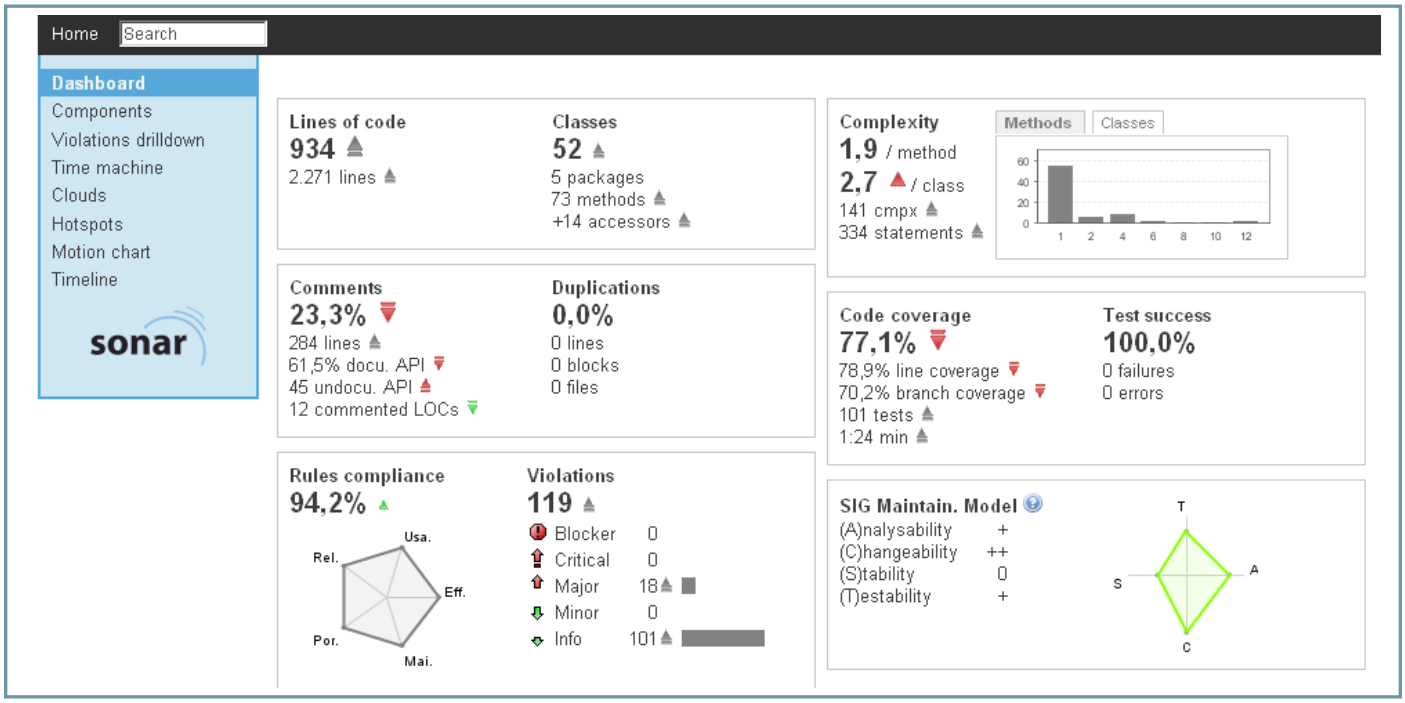
As discussed in the previous section, Hudson allows us to execute the unit tests automatically every day. Furthermore, it allows the plug-in execution to measure the code coverage with several open source plug-ins like “Cobertura” or “EmmaECL”. These plug-ins are executed with the unit test and produce several metrics such as line coverage, branch coverage or the cyclomatic complexity.





Other features you can add in the Continuous Integration System are the code static analysis such as PMD, FindBugs or Checkstyle. These tools provide an analysis of the source code without executing it by making some checks against predefined rules. The result is a collection of metrics which can be used to obtain a quality statement about the source code.

To facilitate the managing of all the metrics and results obtained through the Continuous Integrated System, the Sonar application can be used. Sonar is an open platform to manage code quality and can be connected with Hudson through a plug-in to execute all the code analysis tools and collect the results. The next picture shows the Sonar dashboard with a usable and intuitive interface.



Thus you can obtain every day the following metrics:

- Lines of code, classes, package and method numbers
- Comment lines, documented API
- Dead code
- Duplicated code
- Rules compliance divided in five categories (efficiency, maintainability, portability, reliability and usability) and divided according to their critical nature.
- Test unitary execution results
- Line coverage and branch coverage.
- Class and method complexity
- SIG Maintain Model divided in four categories (analyzability, changeability, stability and testability.
- Quality Index. This index is composed of different metrics (rules violations, complexity, coverage and code style).

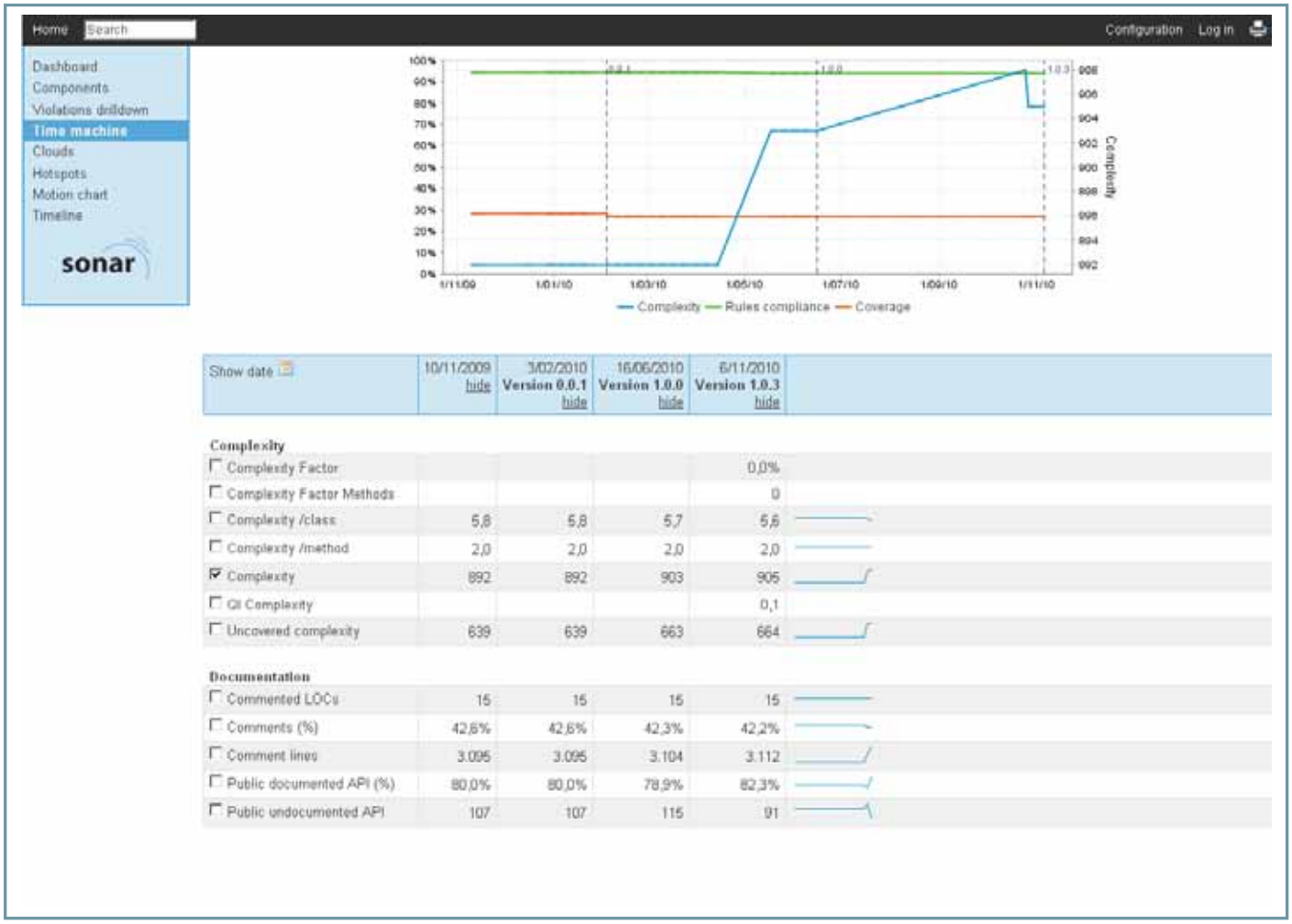
Sonar plug-in also obtains different trends of every metric to analyze the evolution of the software quality. Sonar helps you replay the past and show you how quality metrics evolve in time. Therefore, it makes the actions taken easier in order to improve the quality of the source code.

Why use Hudson + Sonar?

There are different benefits to using the Continuous Integrated System:

1. Early detection of compilation and integration errors because the source code is integrated and built every day.
2. Automatic execution of the unit and integration test. This execution can detect every-day errors introduced in the code and the development department can fix these errors in an early development phase.
3. Automatic execution of the code analyzers. This feature takes a daily photo of the development process state and helps the project manager to take actions to improve the development process and the quality of the product.

All these benefits can be provided by the combination of Hudson + Sonar providing a flexible system which allows all the metrics and rules configurations and provides a usable and intuitive interface to all users.



Biography

Antonio Robres is a Test Engineer at Diagnostic Grifols in Barcelona, Spain. He studied Telecomunications Science at Universidad Politéctica de Cataluña in Spain and has a Master in telecommunication administration and is an ISTQB© certified Tester Foundation Level. He has been working for 5 years in the field of Software Testing and Quality engineering in companies such as Telefonica, Gas Natural and Grifols. His work focuses on the design and execution of several testing projects, mainly in embedded systems and web applications. He is also involved in the design and development of test automation projects with Open Source tools. He was a speaker in the last QA&TEST edition explaining all the testing and QA structure of Diagnostic Grifols in detail.

Populating your database with a few clicks

by José Carréra

A common obstacle frequently faced by the testing team is the challenge of having to check several testing scenarios which require very large amounts of data that could take months to be obtained in a real production environment. Also, depending on the domain of the application, it may just not be possible to have the data made available from production environment due to security or privacy issues. Therefore, quality assurance teams need to find other ways to simulate these scenarios as closely as possible to real environments, whilst at the same time they usually face strict time constraints, which makes their work even harder.

Test engineers have at their disposal some solutions to deal with this matter. However, they are not always easy to use or are not fast enough. A commonly used approach is to create SQL scripts manually using a text editor, which is a tedious task that consumes a lot of time. Another possibility is to use specific tools to aid in this task, allowing the user to quickly and easily generate large amounts of data. However, it is not easy to find a suitable open-source tool that is flexible enough to adapt to any type of scenario and all data types.

A couple of months ago I was introduced to the **Data Generator** tool, which according to the website www.generatedata.com is „a free, open-source script written in JavaScript, PHP and MySQL that lets you quickly generate large volumes of custom data in a variety of formats for use in testing software, populating databases, and scoring with girls". I still haven't discovered how the tool may help you „score with girls", but it sure is a great tool to assist software testing in populating databases.

In this article I will describe the main features of this application, pointing out ways to better use these features based on our previous experience using **Data Generator** in our projects.

First Look and Installation

The first great thing about the **Data Generator** tool is that you can try it out without having to install it. After accessing the application's website www.generatedata.com and clicking on the Generator tab, you will be presented with that same web GUI that you will have if you later on decide to install it locally on your machine. The only constraint is that using this "demonstration"

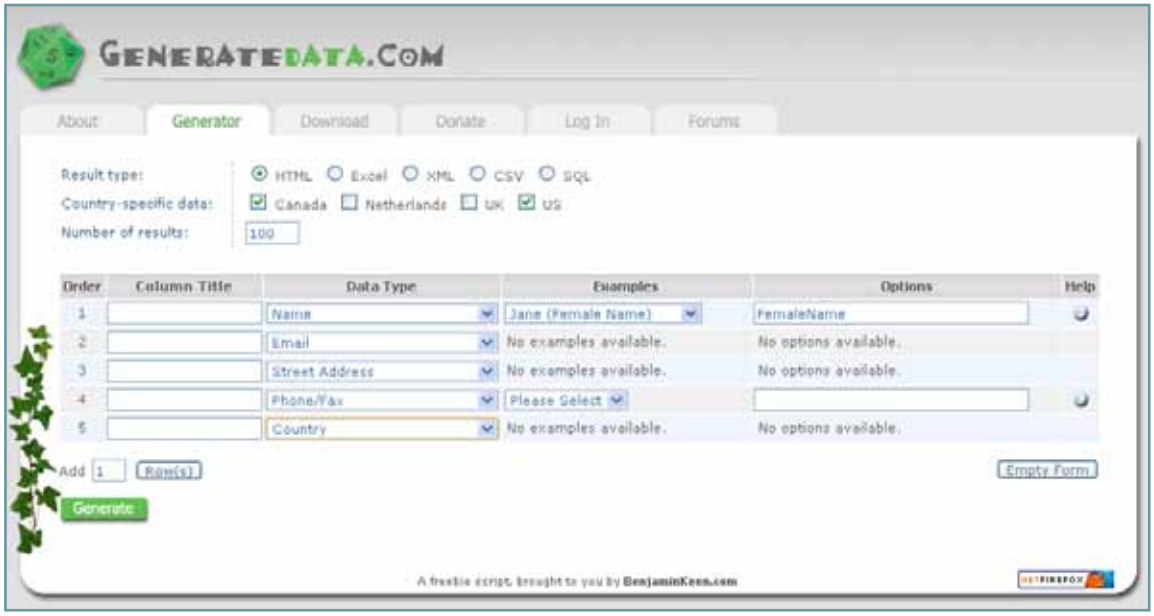


Image 1 - Website Generator version

version you can only create a maximum of 200 records at a time, whereas with the locally installed version you may reach 5000 records at a time.

To install it locally on your machine only a few simple steps need to be performed. By clicking on the download tab at the website you can see what the system requirements are:

- MySQL 4+
- PHP 4+
- Any modern, JS-enabled browser

To fulfill these requirements we installed **WampServer** (www.wampserver.com), another free, open-source project, which allows us to run Data Generator and all its features. After installing WampServer just a couple more steps need to be performed. To start running Data Generator locally on your machine, a simple five- step installation procedure is available via the download tab of the application's website. For **Data Generator** version 2.1, which is the version we used in our projects, the required steps are:

1. Download the zip-file at the top and unzip the contents locally on your computer.
2. In the zip-file, you'll find a file in the /install folder named db_install.sql. This contains all the SQL to create the MySQL tables and raw data used by the Data Generator (names, cities, provinces, states, countries, etc). You will need to execute these statements on your database through any database access tool, such as phpMyAdmin.
3. Edit the global/library.php file. At the top, you'll see a section where you need to enter your MySQL database settings. Note: if you choose to change the database prefix, make sure you rename the tables after running the SQL in #2!
4. Upload all files to your web server.
5. Upload it to your web browser and get to work.

After this small procedure, you can start using **Data Generator**. As we will describe in the next section, it comes with a pre-loaded database that might help you in several test scenarios.

Image 2 - Available data types

Explaining its main features

First, I will describe how we can use the pre-loaded user database to generate data. Almost all applications have a user management feature that uses data like name, phone, e-mail, address, etc. It is also common that at some point during the project life cycle we need to use a larger amount of user data, in order to assess how the system behaves in different test scenarios that may focus on features like response time, data integrity, general user interface, and many others.

On the Data Generator main screen you will be able to define on each row a data type for each column of the table that you want to populate. For each row, the user can define a column title and its data type along with its specific settings. Different types of pre-loaded data types will be available like: name, phone/fax, e-mail, city, and others. These data types allow us to solve different issues, by attending various test scenarios.

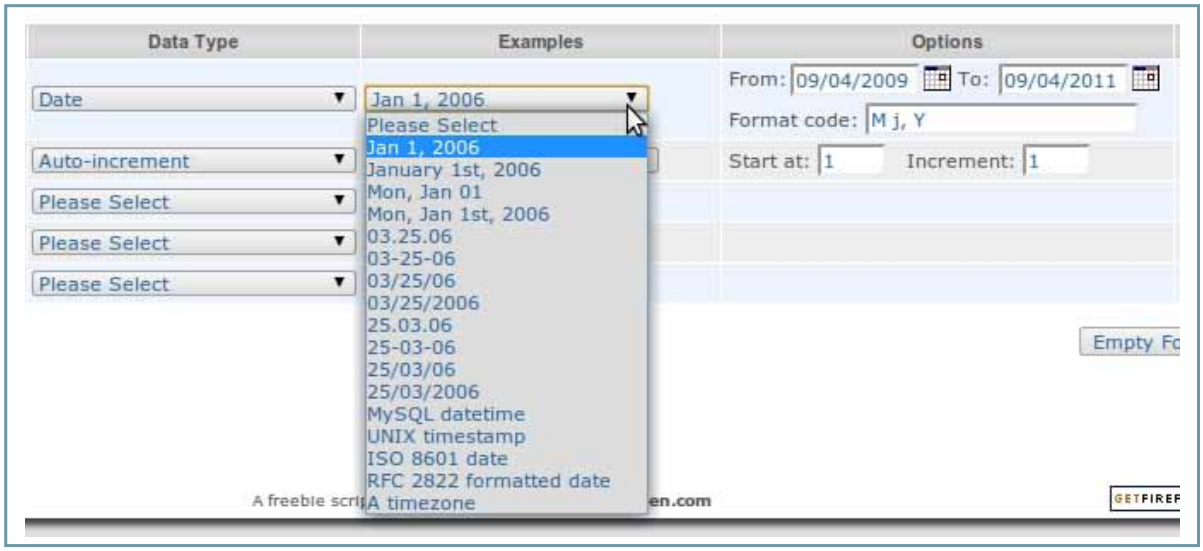


Image 3 - Dates / pre-defined formats

Among the pre-loaded data types, the "Date" option is one of the most useful, presenting a relevant group of variations, as shown in image 3. The "Date" option allows us to solve a common problem faced when generating data, which is that each database management system uses different date formats. With this defined data type, we can solve this issue with a few clicks selecting a date format and setting the desired date interval.

The pre-loaded data can be used in isolation or together with custom data types like alpha-numeric and custom lists, which are two different data types available where the user can enter specific custom values. If you select custom list, you can choose among some pre-loaded available lists like: marital status, colors, titles, company names, etc., or you can define a customized list by entering values separated by a pipe | character. You may also define before generation the number of values that you want to be included in each record.

You can also choose alpha-numeric data types, where you can either define a specific static value which will be repeated for each created record or use its feature that allows the generation of random values. To do so, you need to follow some defined rules presented below in image 4. For instance, if you set the value field to LLLxxLLLxLL, records will be generated replacing the ‚L' for random upper-case letters and the ‚x' for any number from 0 to 9.

Alphanumeric

This data type lets you generate random alpha-numeric strings. The following table contains the character legend for this field.

L	An uppercase **L**etter.	**V**	An uppercase **V**owel.
l	A lowercase **l**etter.	**v**	A lowercase **v**owel.
D	A letter (upper or lower).	**F**	A vowel (upper or lower).
C	An uppercase **C**onsonant.	**x**	Any number, 0-9.
c	A lowercase **c**onsonant.	**X**	Any number, 1-9.
E	A consonant (upper or lower).		

Image 4 - Alpha-numeric data type rules

Another important and useful data type available is the auto-increment, which we can use to generate a unique number for each row by incrementing an initial configured value by whatever value you enter. This functionality is very helpful for setting numeric values commonly used for primary keys on database tables.

Finally, we present the settings that are available for generating records. Initially, the user can choose among the different exporting formats, a feature which has shown to be very useful considering that we might use the generated data in different situations, like using it to import via the application's front-end (if available) or data importing directly via the system's DBMS (Data Base Management System). We can also choose betweem different settings of the pre-loaded database, choosing which country we want the records to be retrieved from, allowing more precise information. Last but not least, we must define the number of results or records that we need to be generated for the selected configuration (as mentioned before, we can reach 5000 records at a time with the locally installed version).

Remember that I am not trying to describe a perfect tool that can solve all your problems. You might find some drawbacks, limitations and also find yourself having to adapt the available data types to your needs. However, that's the great thing about an open-source project; you can help to make it get better by improving its features, adding new ones, fixing bugs or just by exchanging ideas with developers and providing important feedback for future releases.

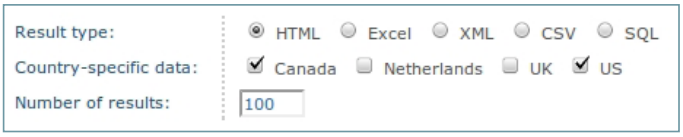


Image 5 - Results settings

Conclusion

Technologies applied on software projects get more complex every day, along with a greater challenge for quality assurance teams to assist the development team in releasing products with higher quality standards, capable of complying with user needs and fulfilling expectations.

To achieve these demands, the usage of tools to assist the work performed by the quality assurance team is primordial. Allowing engineers to provide faster and more accurate results to their projects, **Data Generator** seems to be a great option, providing test teams with a suitable solution for populating databases.

In this article, we presented the main features of **Data Generator**. Special thanks go to Benjamin Keen, responsible for the development and distribution of the tool. For further information regarding this tool or the challenges that it might help solving, please contact the author.

Biography

José Carréra, MSc, is a test engineer at C.E.S.A.R. (Recife Center for Advanced Studies and Systems) since 2006 and Professor of Computer Science at FATEC (Faculdade de Tecnologia de Pernambuco), Brazil, since 2010. He obtained his master degree in software engineering (2009), graduated in computer science (2007), and is a Certified Tester — Foundation Level (CTFL), by the ISTQB® (International Software Testing Qualifications Board - 2009). His main research interests include quality assurance, Agile methodologies, software engineering, performance testing, and exploratory testing.

Metrics can be a mine field

by Eitan Ganor

Quality methodologies, and CMMI® on top, push for quantitative methods in decision making. This is good as long as it is done in a cautious way, yet could lead to disaster if organizations become addicted to the method and not judgmental to what the metrics tell you.

A metrics system is crucial in global companies that have development centers in different countries and often also decentralized projects. The only way to have a common management language is by basing it on a metrics system, otherwise issues will be judged differently in various parts of the company due to differences in culture.

Here are some possible "mines":

Mine 1 – Overflow of data.

Decision making should be based on information, yet a lot of data is not useful information. Too much data, derived from a large number of metrics applied, might bring you many reports with a lot of numbers, yet you will not necessarily know what to do with it.

Design your metrics based on the business targets, in a mapped way, so that there is a reason for each metric. When designing a metrics system, „more" is not „better".

Mine 2 – Contradicting data

When designing a metrics system, special attention should be given to not have coupled metrics that are interpreted in a contradicting way.

I have seen an HR metrics system that judged overtime as positive and project delays as negative. The problem was that when a project is late, people do a lot of overtime, so the two metrics neutralized each other.

Mine 3 – Disregarding the data

People have the tendency to focus on metrics that support their views, and disregard metrics that are in conflict with their views. Managers are no different, so quality managers have often the problem of convincing management to discuss all the metrics results.

Mine 4 – Biased data

Metrics systems should be designed in a balanced way. Each measured issue should have the same weight in the overall system. Having too much metrics in one issue might cause biased information. A good example is the default management systems that provide us with a lot of metrics. We should focus on one metric that coincides with our business targets, and not grab the whole package, which would result in too much weight being given to this subject.

Mine 5 - Becoming enslaved

No metrics should replace common sense. Becoming enslaved to numbers causes non-creative thinking, resulting in management being replaced by artificial intelligence computer system.

Metrics outcome should always go through thorough discussions to ensure that they serve the achievement of business targets.

Mine 6 – Differences in culture in global companies

The hi-tech world is global. Work is spread in many locations in different countries.

Metrics programs should take into consideration the differences in culture and should permit local adaptation.

More than that, a global company's metrics program should be designed specifically to overcome obvious traps. One example is the trend to shift work east, where labor costs are lower - but is the product development cost cheaper? This depends on other factors like efficiency, creativity, quality, etc. So, in global companies the total development cost per product should be measured, rather than hourly costs.

In conclusion, metrics play an important part in managing companies. They create common knowledge DB, and assure that in the decision-making process everything is taken into consideration. Just make sure they are not misleading.

27 years in the hi-tech Industry, of which 20 years in quality management of global companies.
Started as a CAD-CAM software development engineer in aviation industry, continued as a test engineer of aerospace test field computer center and various security systems.
In the next years the roles of software quality engineer and software quality manager were fulfilled.
The next station was building and managing an audit project for a global telecom IT systems' developer. In the last few years the role was of programs' quality manager of the local development center, assimilating methods of a world class global telecom networks development & manufacturing company.

Along these years public nominations were fulfilled, one was as a member in the board of directors of a large public IT company, representing the government, and the second is representing the Israeli Consumer Council in the Standards institute of Israel, in technical committees.

Qt Application Test Automation With TDriver

by Petri Kiiskinen

Background and motivation

Nokia recently announced that they will be focusing on Qt[1] to provide developers a single interface to create applications and services on mobile devices and desktops. Qt will make it possible to create one application that works on mobile devices and desktops alike. Developers can use C++, Javascript & html and/or QML to develop their applications.

Testability Driver (TDriver for short) is a testing tool open sourced by Nokia. It will make test automation possible for Qt applications running on any platform that runs Qt. Platforms we have used include Linux, Windows, Mac, Symbian, maemo and MeeGo.

Use cases for TDriver

Functional UI test automation

TDriver can be used to make tests through the UI layer. Agile software development teams, for example, can include passing UI tests as part of their definition of done. The tested UI can be any Qt UI like QWidgets, QGraphicsView, web content or QML.

Functional test automation through business logic

This kind of testing will bypass the UI layer with fixtures developed by developers and testers together. Fixtures are plug-ins executed on the SUT. These tests are usually faster to execute and more robust – and need less maintenance, especially if the UI changes a lot.

Rendering testing

TDriver can use Imagemagick libraries to do rendering verification through screen captures and bitmap verification.

Multiple device testing

It is possible to control multiple devices in one test. These kinds of scenario include multiple mobile devices communicating with each other. Also, one scenario might be to control a PC through a Qt based web browser accessing sites, and then switching to mobile device and browse the same sites or use dedicated applications that access those services.

Localization testing

The user can generate a database containing translations for their application. Using this information it is possible to verify that correct translations are in place when switching languages.

Performance testing

TDriver can be used to make performance tests from the user experience point of view. This means high-level measurement points which are easily measured and can be used to test regression. It is possible to monitor events and signals of the application and use this information for measuring performance, e.g. application startup time or end-to-end performance of any use case. Effective performance testing will require instrumentation of the applications, e.g. placing proper signals into measurement points.

Reliability testing

Reliability testing runs a set of test cases multiple times. Using TDRunner, the execution of these test cases can be randomized and one can give importance to certain test cases (weights) if they need to be executed more than others. TDRunner can be set to run a for a number of iterations, a number of hours or until a predefined date & time. Reliability testing is usually executed for days or even weeks. TDRunner uses TDriverReporter which is a HTML-based report that can be written to disk in real-time. This way it is possible to interpret a test case result immediately after it has been executed, even though the suite is still ongoing.

Memory consumption monitoring

During the functional tests it is possible to monitor memory consumption of the SUT or application. Simple memory leaks are easily detected if the memory consumption trend is growing. If regression is detecte,d then it is possible to execute tests under Valgrind[2] for a more detailed analysis.

Continuous verification

It is possible to configure additional verifications that will take place continuously in the background. For example, one could measure that the application memory consumption does not exceed a certain threshold.

Using webcam to store video

During testing it is possible to record a video from SUT using webcams. It can be configured to store only the failing case and the

HANDBUCH

TESTEN
IN DER FINANZWELT

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

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

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

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

Herausgegeben von Norbert Bochynek und José M. Díaz Delgado

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

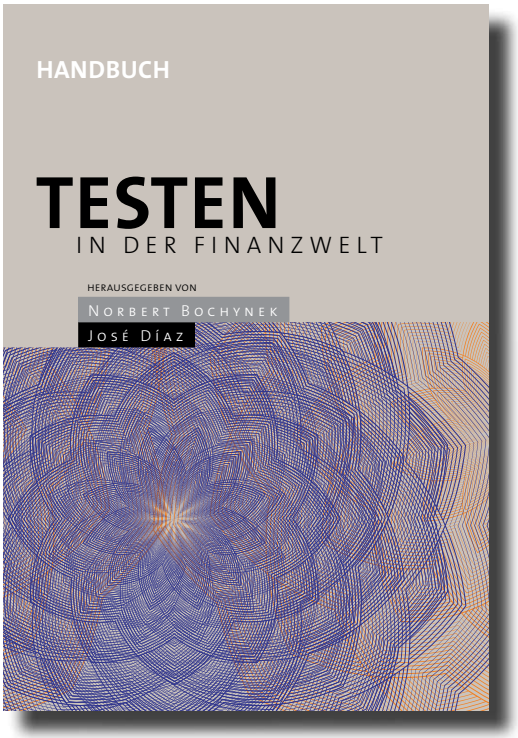
Gebundene Ausgabe: 431 Seiten

ISBN 978-3-00-028082-5

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

48,00 € (inkl. Mwst.)

www.diazhilterscheid.de

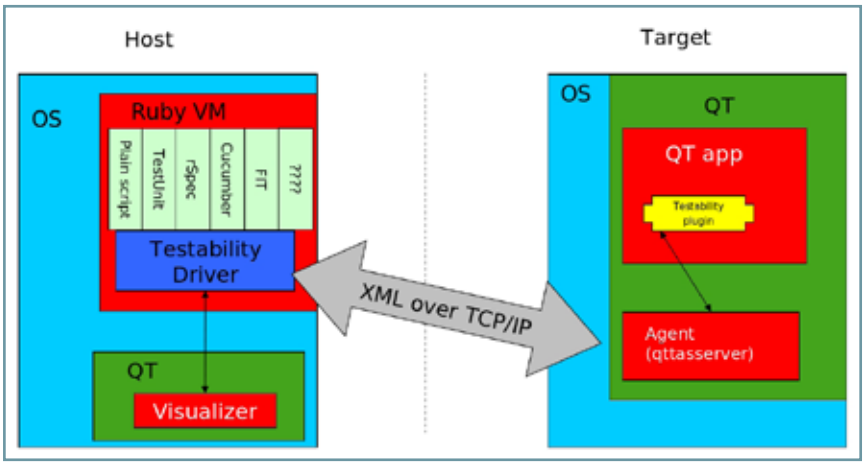


case before that.

Generic reporter

All test harnesses provide their own style for reporting. TDriver also provides a generic reporter which can be used without a harness or together with xUnit or Cucumber harnesses. The report will store all the results and provide statistics on the run, and it can be made to draw graphs on the performance of test cases. For failures the report will capture screenshots, provide the exceptions that occurred and attach a video if the webcam was configured.

Basic architecture



Language

Ruby language[3] was selected as the scripting language for TDriver. Ruby is very expressive and easy to learn.

Agent

The agent component runs on the SUT and will handle the communication between the applications and the testing framework.

Testability plug-in

This library is loaded by started applications. This will give access into processes of the tested application.

Communication

Agent communicates with testing frameworks using XML over TCP/IP.

TDriver ruby library

This is a class library implemented in Ruby language and provides access to communicate with the target SUT in Ruby language.

Testing harnesses

One can use any testing harness that exists for Ruby, or one can choose not to use any harness at all.

Visualizer

Visualizer is an application that visualizes the application under test. It will help testers to find out the objects in the application and the properties for each object. Visualizer also contains a simple code editor with syntax highlighting and debugging functionalities. Using Visualizer, the user can hover on top of the image and Visualizer will search for the object in those coordinates and highlight the object in the object tree. The user can then right-click the object, and then select 'insert to editor' in order to script using the object.

Choosing the right harness

xUnit

Test::Unit[4] is the xUnit harness for Ruby. Minitest [5] can also be used. xUnit harnesses are familiar to many people, so it might be a natural choice to start with.

FIT

Framework for Integrated Testing [6] was one of the first test harnesses for Agile software developers. It uses a use case idea, and input arguments and results are provided in a table.

rSpec

Behavior Driven Development is said to be what TDD was supposed to be in the first place. rSpec[7] is not too much different from xUnit frameworks, but the language used is about the supposed behavior of software modules.

Cucumber

Recently, the test automation community has started to talk about Acceptance Test Driven Development. This also concentrates on the high-level behavior of the application. Cucumber[8] is a harness that provides almost natural language to define requirements for features to be implemented and examples on how the feature should work in different situations. These examples (scenarios) are also used as test cases. Cucumber provides a nice way to raise the level of abstraction and provides a way to do test automation without scripting. Those who are more comfortable with scripting can create robust step definitions, whilst the rest can concentrate on creating test cases with zero scripting involved.

Example of Cucumber feature and scenarios:

```
Feature: Calculator - calculations
  In order to trust the calculator
  As a User of Calculator
  I want the calculations to return correct answer

  Scenario: 1+1
  Given I launch application "calculator"
  When I calculate "1;+;1"
  Then The calculator display says "2"

  Scenario: Division by zero
  Given I launch application "calculator"
  When I calculate "1;÷;0"
  Then The calculator display says "####"
```

An implementation for one step "I calculate", which can be reused in numerous scenarios as above:

```
When /^I calculate "([^\"]*)"$/ do |calculation|
  calculation.split(';').each do |button|
    @app.Button(:text => button).tap
  end
  @app.Button(:text => '=').tap
end
```

Summary

TDriver is already a very powerful testing environment. However, there is always scope for improvement. We look for community contributions into TDriver. Interesting development ideas might include:

- implement Cucumber scenarios in Qt C++ or Javascript
- implement test case skeleton recorder
- platform specific fixtures for operations on the SUT
- Python support for scripting.

Where to get more information on TDriver?

Sources: http://gitorious.org/tdriver

Project: http://projects.forum.nokia.com/Testabilitydriver

MeeGo wiki: http://wiki.meego.com/TDriver

References

1. http://qt.nokia.com/
2. http://valgrind.org/
3. http://www.ruby-lang.org/
4. http://rubyforge.org/projects/test-unit/
5. http://bfts.rubyforge.org/minitest/
6. http://fit.rubyforge.org/
7. http://rspec.info/
8. http://cukes.info/

Biography

Petri Kiiskinen has been a Managing Director at Jidoka Technologies for three years. He has worked in implementing test automation tools and solutions for the past 8 years, in the early years in the area of commercial tools and solutions, but the last three years in implementing open source based test automation. Prior to this, he worked for almost ten years as project manager, architect, and developer in numerous projects implementing solutions with MS, Java and mobile technologies.

Test automation with TestLink and Hudson

by Bruno P. Kinoshita & Anderson dos Santos

Lately test automation has been discussed by many authors, especially those interested in agile methodologies like Extreme Programming (XP) where Test Driven Development is a key feature to produce better-quality software. There are many tools for Test Automation like HP Unified Functional Testing, Rational Robot, TestComplete, among others. What we discuss in this article is not another test automation tool, but a way of automating your tests using open-source tools that your company may already be using.

In the test automation solution shown in this article, we use TestLink to manage test planning and manual executions (features of the vanilla product) and extend it to handle automated tests. This is done using Hudson which is responsible for triggering scripts, scheduling jobs and dealing with parallel job execution.

TestLink, as stated on the project site, "is a web-based test management tool. The application provides test specification, test plans and executions, reporting, requirements specification and collaborates with well-known bug trackers". TestLink has also an embedded XML-RPC API that can be accessed through a HTTP URL. TestLink was originally developed by Chad Rosen and is now maintained by an active community of developers and testers led by Francisco Mancardi, Andreas Morsing and Martin Havlát. The code is written in PHP and licensed under the GNU Public License.

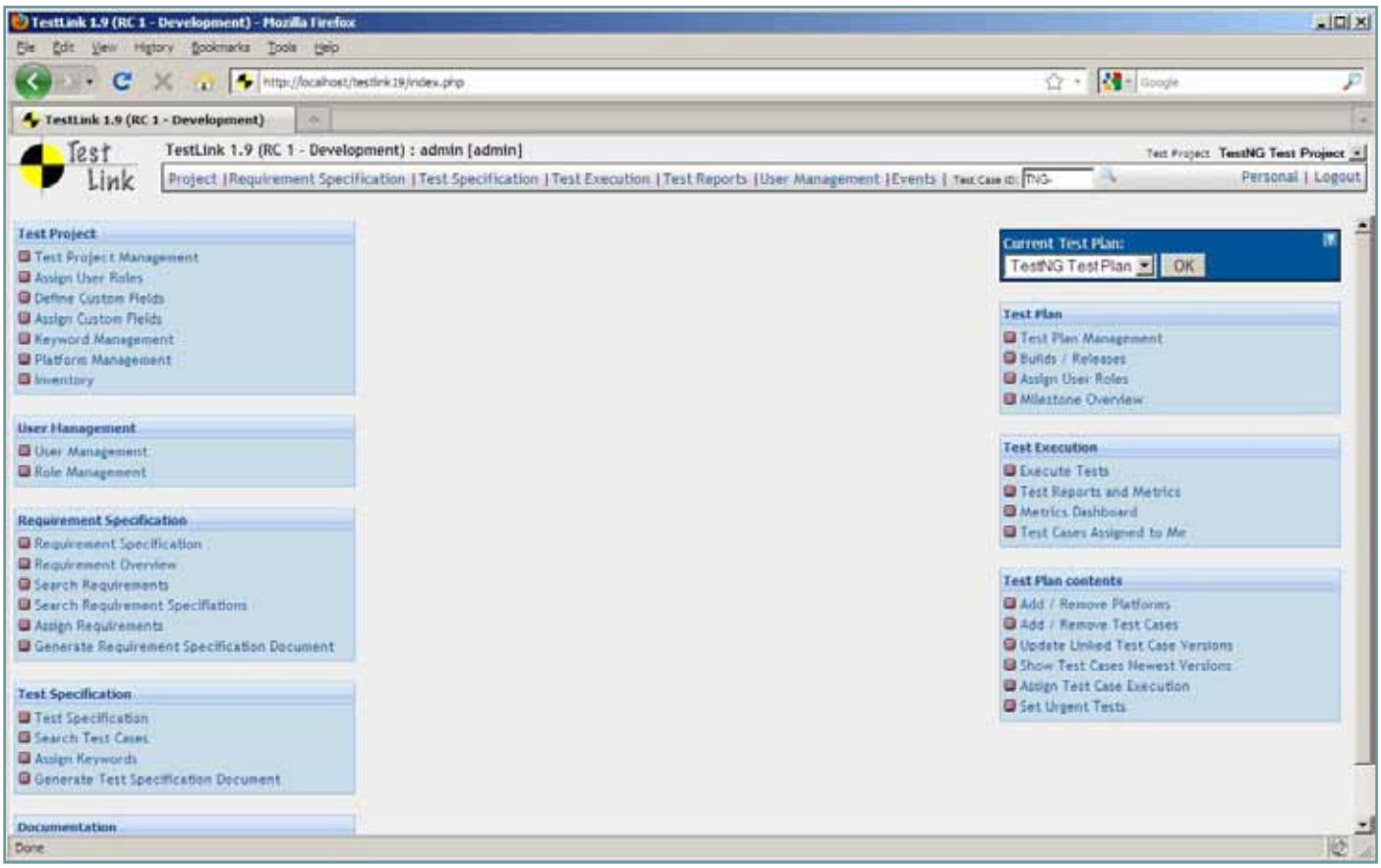


Figure 1 – TestLink 1.9 user interface

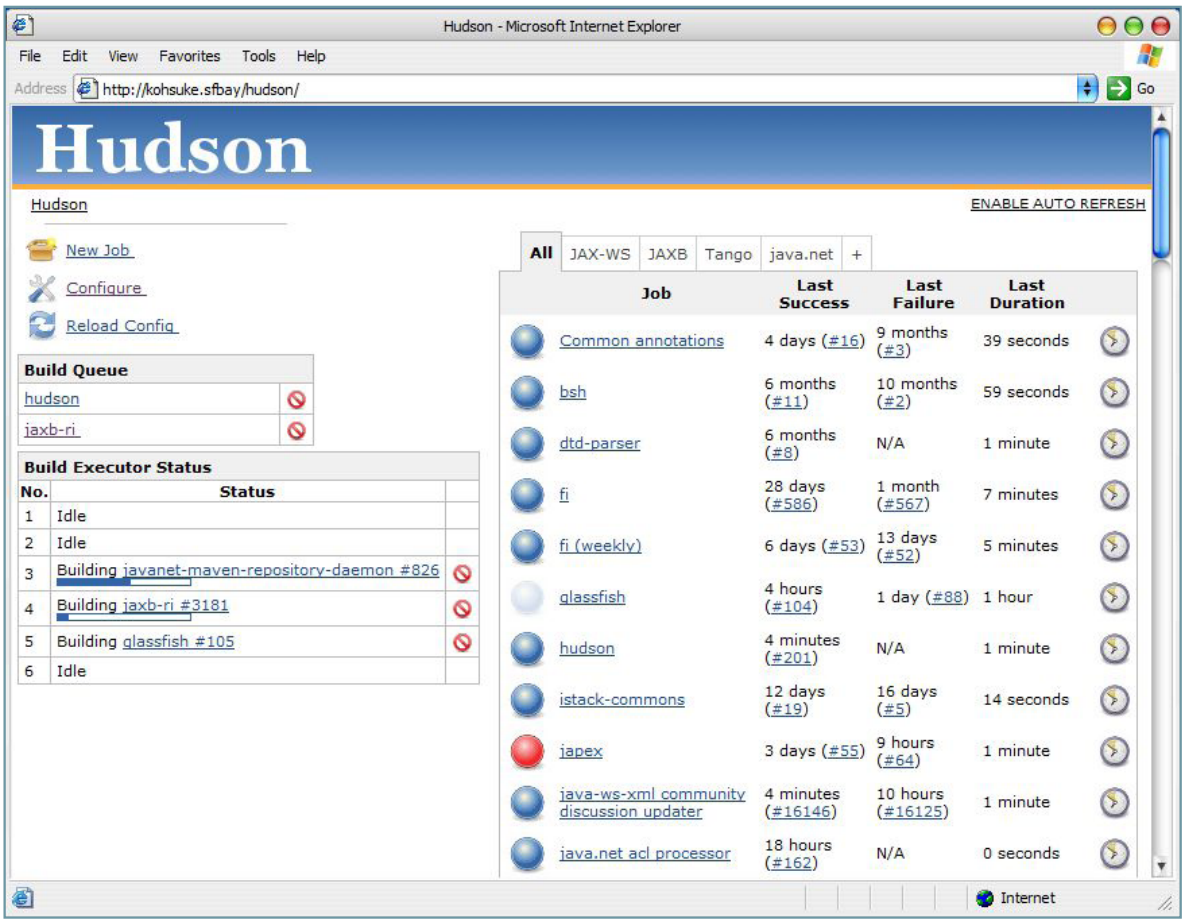


Figure 2 – Hudson user interface (source: Hudson Wiki - http://wiki.hudson-ci.org/display/HUDSON/Meet+Hudson)

Created by Kohsuke Kawaguchi while he was working for Sun Microsystems, Hudson is an open-source Java Continuous Integration Tool that lets you configure jobs, build them and analyze the results. Each job can have its own configuration, which includes commands and plug-ins to invoke source code repositories to download, schedule time and other useful options.

Hudson is very easy to install and configure and contains many ready-to-use plug-ins that allow you extend it. The Hudson community is very active and supportive so even without commercial support you can still feel confident about the solution.

As you can see, neither TestLink nor Hudson are test automation tools. However, being used together in conjunction they can complement each other and help you to manage and automate your tests. Keeping this in mind we developed a plug-in for TestLink and Hudson, the TestLink Hudson Plug-in.

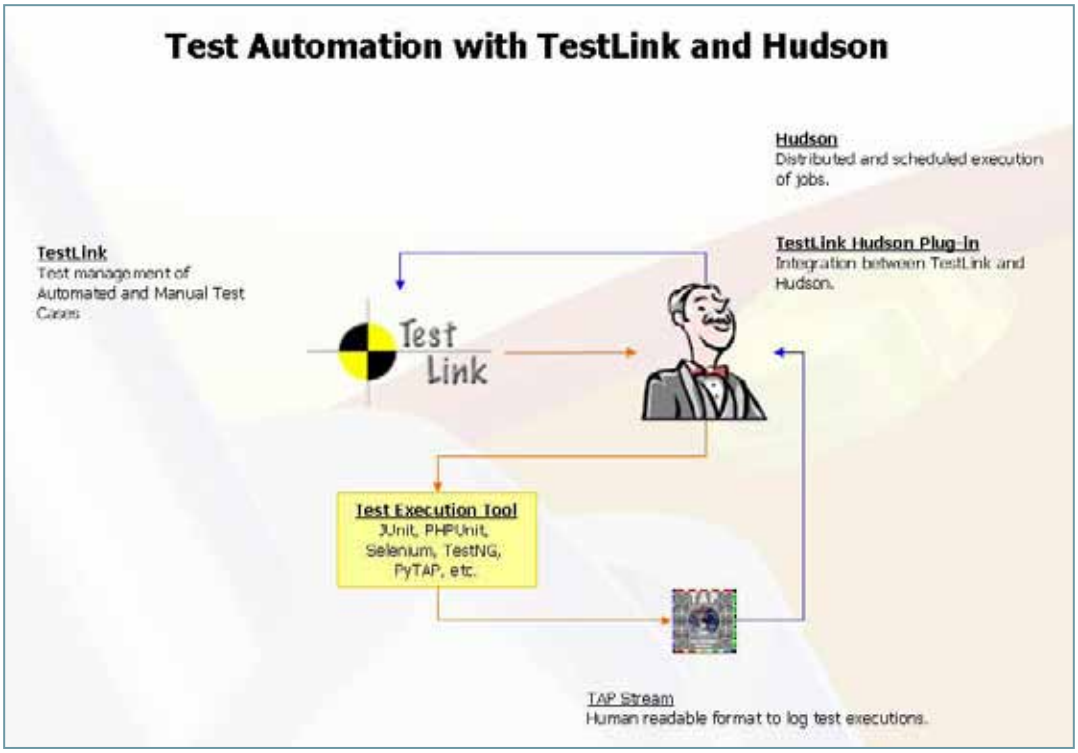


Figure 3 – TestLink Hudson Plug-in 2.0

Subscribe for the printed issue!

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

Billing Adress

Company: _____

VAT ID: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

Delivery Address (if differs from the one above)

Company: _____

First Name: _____

Last Name: _____

Street: _____

Post Code: _____

City, State: _____

Country: _____

Phone/Fax: _____

E-mail: _____

Remarks: _____

1 year subscription

☐ **32,- €**
(plus VAT & shipping)

2 years subscription

☐ **60,- €**
(plus VAT & shipping)

Date

Signature, Company Stamp

TestLink Hudson Plug-in downloads automated test case information from TestLink and calls a test executor tool to run the tests for you. It then checks the results of the test execution and updates TestLink. Hudson has the control over what was executed, when it was executed, how long it lasted, who started or scheduled the execution, and so on. Test engineers and testers will benefit from this information, whilst test managers will pay more attention to TestLink execution results and reports, either manual or automated.

TestLink Hudson Plug-in version 1.0

In the first version of the plug-in we used Maven as a test execution tool. So in order to use this version you have to create a Maven project with either JUnit or TestNG. The plug-in downloads information about a test case from TestLink and calls Maven passing a file name that is specified in a custom field created in TestLink. This file name refers to a JUnit or TestNG test class.

TestLink Hudson Plug-in 1.0 requires TestLink version 1.8.x and Hudson version 1.366 or higher.

TestLink Hudson Plug-in version 2.0

The release of the plug-in's new version is expected for November of this year. The greatest change in this new version is that you will not need to use Maven anymore. Instead of invoking Maven for each test and checking its execution result, we let you specify a custom test command and give you all the information retrieved from TestLink as environment variables.

An important point is that your command ought to output TAP files. Instead of making you dependent on Maven and limited to Java language, you can use any tool in any language that can output TAP. Doing it in this way allows you to use TestNG, PHPUnit, Perl Test Harness, or any other test command that outputs in TAP.

TAP is a protocol for tests. It is an acronym for Test Anything Protocol. It is a simple text-based and human readable open standard that registers test results. The following is an example of a test result in TAP.

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid.
More output from test 2. There can be
arbitrary number of lines for any output
so long as there is at least some kind
of whitespace at beginning of line.
ok 3 - Read the rest of the file
#TAP meta information
not ok 4 - Summarized correctly # TODO Not writ-
ten yet
```

Listing 1 – Sample TAP output (source: Wikipedia article on Test Anything - http://en.wikipedia.org/wiki/Test_Anything_Protocol)

The plug-in uses tap4j to generate and parse TAP files. tap4j is an implementation of the TAP protocol written in Java by us and can be used with other known Java Test Frameworks such as TestNG and JUnit. TAP has implementations in C, C++, Python, PHP, Perl, Java and others. For a complete list of TAP parsers and producers, check the protocol project site in the References section at the end of this article.

In version 2.0 of the plug-in, we also replaced the old TestLink Java API, dbfacade-testlink-rpc-api, with testlink-java-api that enabled us to upload test evidences after executing the tests.

TestLink Hudson Plug-in 2.0 will be compatible with TestLink version 1.9.x and Hudson version 1.366 or higher.

What did my company gain with test automation being done with TestLink and Hudson?

Firstly, I don't need to mention that my company saved quite a lot of money using the TestLink Hudson Plug-in rather than purchasing another tool and having to train the users and set up the entire infrastructure required for this new tool.

Besides that, in order to automate the test cases we had to first review every test case and decide what was worth automating. In doing this work, we found some errors in the test structure and in the way that many tests were designed.

We are still measuring the outcome of test automation, but from what we can see so far is that test automation is reducing the number of repetitive tasks that our test team has to perform and increasing our confidence in our automated test solution.

Okay, and what about the open-source community?

We feel very happy that we can give something back to the open-source community. Until now, our test automation project has produced the TestLink Hudson Plug-in, tap4j, testlink-java-api and contribution code that was merged into TestLink CVS trunk. And very recently one of the authors of this article was accepted as a member in the TestLink team, so more contributions are on the way.

Final notes

Using existing tools to automate your tests not only reduces the budget required to automate your tests but also the amount of problems that you have to overcome when acquiring a new tool, like training, licensing and infrastructure, to mention just a few.

It is also important to note that it will take a lot of effort to develop and maintain test automation and it is a good idea analyze thoroughly what really makes sense to automate and the associated costs and benefits. Just employing a tool won't guarantee that you will achieve success automating your tests. You have to keep in mind that there are other important factors to consider when it comes to test automation, such as the skills required by your team, the complexity of your system, the costs incurred and what your expectations are.

The contents of this article was presented at Encontro Ágil 2010 (an Agile meeting held in São Paulo in November) as a talk. We are waiting for the material to be assembled to upload it and make it available as short tutorials, screen casts and online presentations.

While we are working to finish the TestLink Hudson Plug-in 2.0, we are already thinking of ways to measure code and requirements coverage. As our company already uses Sonar quality dashboard, we are looking in that direction. We imagine that we can publish results, perhaps weekly, about test executions with its code and requirements coverage. We would be delighted to see more people working on these projects (TestLink, TestLink Hudson Plug-in, Hudson, tap4j and testlink-java-api) or contributing with suggestions. If you are an experienced developer, tester or someone with great desire to help, please do not hesitate to contact us. And do not miss the forthcoming version of the TestLink Hudson Plug-in.

References

Linked-in Discussion about Test Automation - http://www.linkedin.com/groupItem?view=&gid=86204&type=member&item=30438098&qid=3d61830a-18e6-49b5-b766-be46cba6373e&goback=.gmp_86204

TestLink - http://www.teamst.org

Hudson - http://www.hudson-ci.org

Hudson Wiki - http://wiki.hudson-ci.org

TestLink Hudson Plug-in - http://wiki.hudson-ci.org/display/HUDSON/TestLink+Plugin.

TestLink Java API – http://sourceforge.net/projects/testlinkjavaapi

tap4j - http://tap4j.sourceforge.net.

Test Anything Protocol – http://testanything.org/wiki/index.php/Main_Page

Wikipedia article on Test Anything Protocol - http://en.wikipedia.org/wiki/Test_Anything_Protocol

Sonar - http://www.sonarsource.org.

Bruno P. Kinoshita – http://www.kinoshita.eti.br

Encontro Ágil 2010 - http://www.encontroagil.com.br.

Foundations of Software Testing ISTQB Certification.

Biography

Bruno P. Kinoshita lives in Sao Paulo in Brazil. He currently works as consultant with Sysmap Solutions (www.sysmap.com.br). Bruno has 7 years commercial IT experience, mainly in telecom services, with Claro, Tim and Vivo (mobile carriers from Brazil). His experience covers Java development, requirements analysis, system architecture, test management and quality analysis. Bruno holds SCJP, SCWCD, ITIL and Cobit certifications and is member of various open-source projects such as Hudson, TestLink and Crux. You can find more about Bruno P. Kinoshita through his personal website www.kinoshita.eti.br.com.

Anderson dos Santos is an Enterprise System Solution Architect with vast experience in telecommunication environments. He has years of experience and strong knowledge in implementing CRM suites and integrations between front-end solutions, billing systems, ERP and legacy systems (mainframe, home-made, etc) using SOA approach. He currently works with Sysmap Solutions (www.sysmap.com.br) as Senior Software Quality Director. Find more about Anderson through his personal website andersonxp.tumblr.com.

Es gibt nur einen Weg zum Glück.

gebrauchtwagen.de
Autos zum Verlieben.

QAliber Test builder: Suited for System Testing in an Agile Environment?

by Erik Melssen

This article discusses my own practical experience using the open-source tool QAliber Test builder for automating a set of system tests. QAliber is a lightweight test automation framework. If you are planning to start with simple test automation or want to know about a user's experience with QAliber, this article will be of interest to you.

My selection process

While working on a project for a non-profit organization with limited financial resources, I needed a cheap or free tool to perform simple automation tasks for endurance testing. As not only my budget but also the available time were limited, the tool needed to be lean. Many of the proprietary tools seemed to be huge and would take weeks or months to get acquainted with, or were simply too expensive for the foundation I was working for. This forced me to look into open-source tools. Many of these were dedicated to web testing and would not fit my purpose. For an overview of what's availabe, see www.opensourcetesting.org/functional.php and en.wikipedia.org/wiki/List_of_GUI_testing_tools. I needed a tool that could drive an application GUI to control the connected embedded hardware. From the tools that remained, QAliber test builder seemed to be the most promising for performing my functional endurance test from a Windows environment.

My learning process

Downloading (sourceforge.net/projects/qaliber) and installing the tool was easy and went without noticeable issues. Learning about the capabilities of the tool was also great fun, as the examples are easy and you quickly have a simple test automated. There are some videos available and the project's Wiki is ideal to get you started really quickly. I was able to automate my first intended test within a day, taking into consideration that I have experience with other GUI test automation tools. The environment is very clear and simple. The built-in logging facility logs the results for each test run and enables you to specify which data you want to capture. The built-in test case repository contains all the basic actions you need for building a simple test. The most important are: GUI actions (e.g. mouse, keyboard and window control), system actions (e.g. file actions, process actions) and flow control. The latter is used to create "while" loops, conditional actions and the very important try/recover actions. Test cases can be grouped into categories and subcategories. These categories can be selected and deselected for execution at the press of a button.

My usage process

While using the tool I ran into some issues. Firstly the tool is not very stable yet. When I ran the tool for a long time, it crashed from time to time. Furthermore, the tool seems to have troubles identifying my slider controls, which is something that could be worked around. Also, it had trouble with a tab control, in which case the tool sometimes selects another tab by accident. As with many of the more complicated tools, the tool lacks an important facility, which is the possibility to create subscripts that can be reused and called with input parameters. If one needs to perform identical sequences of actions and checks in multiple test cases, the data has to be copied and maintained in several places. This is of course a burden and negatively affects maintainability. Also QAliber has problems running tests at full speed, even if you set the option "Delay after action" to its minimal value.

The tool was agile in a sense that it was simple and can rapidly deliver a working test. The non-agile part of the tool is that it cannot be driven from the command line yet, so including it in the automatic build process for continuous integration is not easy. In my opinion this would be a nice extension to the tool so it could be used in a similar way as, for instance, a tool such as Robot Framework. This would make it possible to include it into your build environment and design a functional smoke test to be run each time the build is performed.

What's in it for you

QAliber is a light-weight test automation framework. It contains the basics parts that a test framework should support. It is not keyword or data driven; this would only make the tool complex and is not needed for a simple or intermediate automation task. If you want to perform more serious test automation, you will probably need a tool that can do more. If you want to start system test automation in a small gile project, this tool might be perfect to give you a first insight in the rather complex world of test automation frameworks, especially if you are a tester without development experience and have not lot of experience with automatic testing.

However, there is more. With the QAliber test developer part of the product, test automation engineers can add tests to the framework which a tester can easily select from the test builder. It is possible to use variables and/or lists, which makes the testing quasi data-driven. This part of the product I did not experience in detail (yet), but this has great potential.

My conclusion

Using QAliber to automate parts of my system endurance tests was simple, fun and agile. The fact that I was able to create a couple of basic test within a day was definitely worth the effort. It was almost perfect for the job I needed to perform, creating a couple of flexible system endurance tests that needed to run for long periods of time. If you need to do more complicated automation tasks or need to create a large test set, you will run into its limitations. However, note that development of the tool is still ongoing. Before spending your time and money on a proprietary GUI automation tool, like for instance Automation Anywhere, it is wise to check if QAliber could do the job for you. This might not only save you money but also time.

Biography

Erik Melssen is a Test Designer at Topic Embedded Systems with 14 years of development and test experience for high-tech systems. During the past 9 years of his career he has been in software test (automation) related functions from test engineer to test lead. Erik holds a Masters degree in Informatics and is an ISEB Practitioner.

A brief introduction to the open-source test management system: TestLink

by Terry Zuo

About TestLink

TestLink is a web-based test management and **one of the best QA open-source tools** available.

The tool includes reporting and requirements tracking and co-operates with well-known bug tracking systems, such as Bugzilla, JIRA, Mantis and Test Track Pro from Seapine, Inc.

We've used this system for many years in various large projects, even in a project team of more than 20 engineers and over 1000 cases.

You can use the Test Link during the whole test lifecycle in the project activities.

The current release version is 1.9.

Main features of TestLink:

- The products are tested in test plans, in accordance to standard testing procedures. See IEEE 829, which is compatible with ISTQB terms.
- Users have defined roles (for example: leader, tester, guest).
- Test cases are organized in an hierarchical structure and hold traceable history data.
- Keyword support, allowing greater depth in test organization
- Testing prioritization, tester-specific assignment, and milestone definitions
- Reporting and test metrics
- Documentation export to HTML, MS Word and MS Excel
- Direct e-mailing of reports from the tool
- Localization and internationalization (into English, French, German, Italian, Spanish, Brazilian Portuguese, Polish, Czech, etc.)
- Direct collaboration with bug tracking systems
- Requirements-based testing
- SOAP API for collaboration with functional testing tools.
-

Basic Terminology

- ***Test case*** describes a testing task via steps (actions, scenarios) and expected results. Test cases are the fundamental elements of TestLink.
- ***Test suite*** (test case suite) organizes test cases into units. It structures the test specification into logical parts.
- ***Test plan*** is created when you want to execute test cases. Test plans can be made up of the test cases from the current test project. The test plan includes builds, milestones, user assignments and test results.
- ***Test project*** is a permanent entity in TestLink. The test project will undergo many different versions throughout its lifetime. The test project includes the test specification with test cases, requirements and keywords. The users within the project have defined roles.
- ***User***: Each TestLink user has a role that defines the available TestLink features.

Testlink offers **integration interfaces to different defect control systems**, such as Bugzilla, Mantis, Jira. This way, each defect found can be linked directly to the respective test case and become easily accessible. Testlink also offers a report for the number of defects found in each case, as well as detailed information about each of the problems found.

Case1: Enable the email function as below

(modify the config.inc.php file):

```
/* [SMTP] */
$g_smtp_host = ‚sampleserver.corp.dresser.com‘; #
SMTP server MUST BE configured
# Configure using custom_config.inc.php
$g_tl_admin_email = ‚terry.zuo@dresser.com‘; # for
problem/error notification
$g_from_email = ‚TestLink.admin@dresser.com‘; #
email sender
$g_return_path_email = ‚terry.zuo@dresser.com‘;
// Configure only if SMTP server requires authenti-
cation
$g_smtp_username = ‚terry.zuo‘; # user
$g_smtp_password = ‚password‘; # password
```

Case2: Customized the bug interface system with Mantis:

Herethewebserver(XAMPP)isrunningonhttp://10.40.160.9:8600/

Step1: Enable the /* [Bug Tracking systems] */ in the config.inc.php file as below:

```
/$g_interface_bugs = ,NO';
$g_interface_bugs = ,MANTIS';
```

Step2: Modify the cfg/mantis.cfg.php as below:

```
*/
define(,BUG_TRACK_HREF', „http://10.40.160.9:8600/
mantis/view.php?id=");
/** link to the bugtracking system, for entering
new bugs */
define(,BUG_TRACK_ENTER_BUG_
HREF',"http://10.40.160.9:8600/mantis/");
```

Shown below is the main page after the TestLink system has been set up:

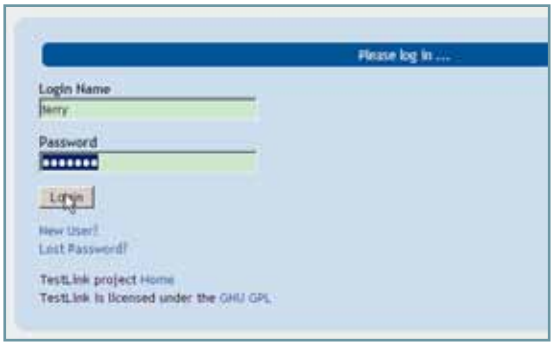


Figure 0,0 - Log-in page

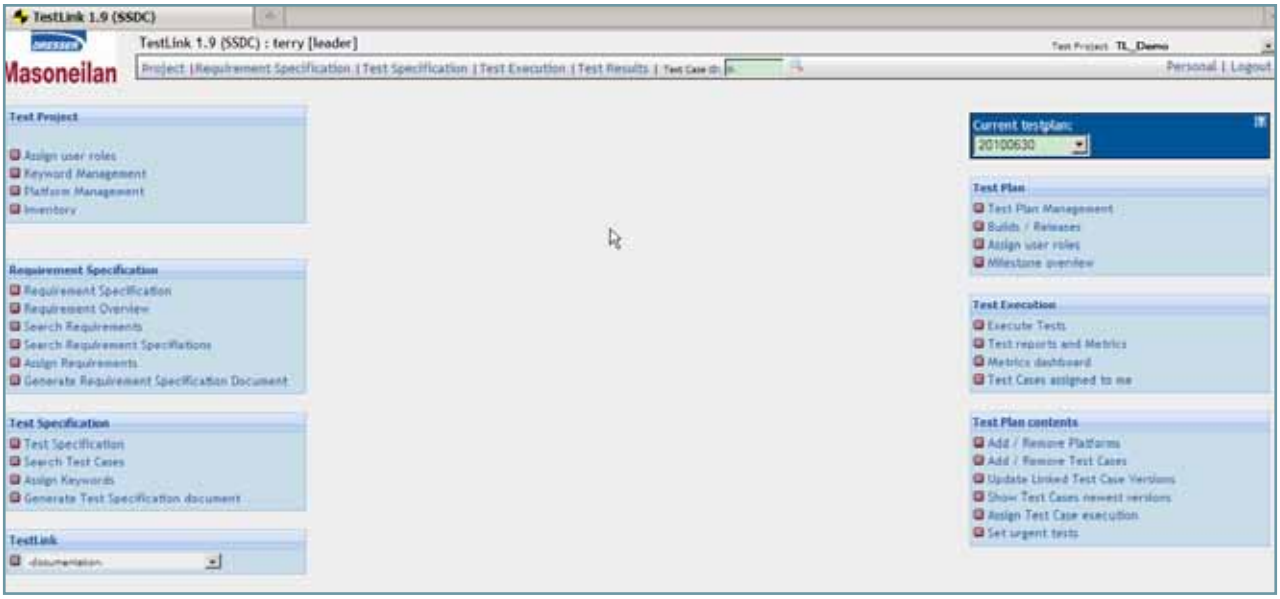


Figure 0,1 - Main Page of TestLink

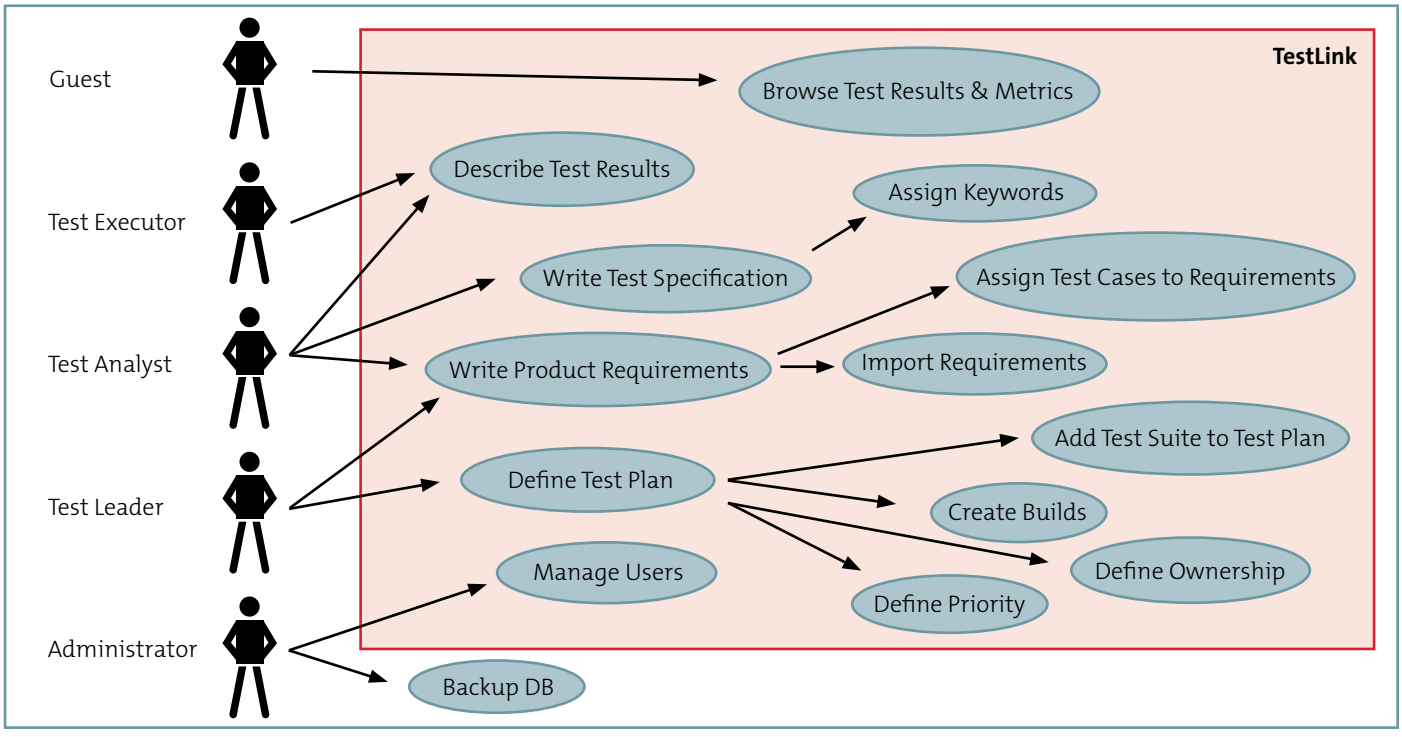


Figure1 - Role map

German Testing Board

www.german-testing-board.info

ISTQB® Certified Tester

Über 145.000 Certified Tester weltweit. Wann gehören Sie dazu?

Manche Ausbildungen öffnen Wege, andere eine ganze Welt.

Die Schulung zum ISTQB® Certified Tester führt zu einem weltweit anerkannten und etablierten Zertifikat. Rund 14.000 Fachkräfte alleine in Deutschland haben es schon. Und Zehntausende in weiteren 47 Ländern - von A wie Amerika bis V wie Vietnam.

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

Anmelden ist einfach.

Ein akkreditierter Trainingsanbieter ist sicher auch in Ihrer Nähe:

GTB Premium Partner

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

autorisierte Zertifizierungsstellen

DLGI - Dienstleistungsgesellschaft für Informatik mbH

iSQI - International Software Quality Institute GmbH

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

Why should we use TestLink

During the software testing routine activities, we have a chance to research advanced and effective tools for test management to handle the increasing test tasks.

Background

According to our internal Quality Management System, and in particular its section concerning testing activities, we had to describe test cases and enter from their execution results in Word & Excel formats as these were suitable and easily understandable for our customers.

The maintenance of these documents was taking up a significant amount of the time for testing activities on the project.

Goal

The research goal was to find a suitable system, which allows the software test engineers to create, perform, manage test cases/scenarios in a centralized repository, and to generate reports from the execution of each test in the same comprehensive way as our Excel template did.

Result

We finally chose Testlink for the easy-to-use and powerful, portable features.

A typical workflow has the following 4 steps:

1. **The first step after you log into the system is to create a test project**. The system allows editing and deleting of projects. If you do not see your test project displayed when you log into the system, make sure you have the appropriate project user role assigned.
2. After you already have a project in TestLink, **you need to create a test plan**. The system offers editing, erasing and deactivating of test plans and test plan versioning. The test plan can be created from an already existing test plan in the project. An option for assigning test scenarios for execution from different users is offered. All test scenarios included in the test plan can be saved in MS Word or HTML format.
3. **Import the requirements** into the system and generate the desired test cases, making sure of the test coverage.
4. After you have created the test plan and the test cases, the next step is to create builds (equivalent to the versions of your development builds). Once your build is created, you are ready to execute all test cases assigned to the current test plan.

Test Execution:

Test execution is available after:

- A test specification has been written.
- A test plan has been created.
- Test cases have been added to the test plan.
- At least one build has been created.
- Testers have appropriate rights for execution of work with the test plan.

Define a tested build

Users should specify one of all the active builds to add results.

The latest build is set by default.

The build label specifies the exact package of the product under test for tracking purposes. Each test case may be run several times per build. However, it's common practice that just one testing round is executed against a build for a test case.

Builds can be created by the test leader using the 'Create New Build' page.

Typical functionalities

TestLink offers an opportunity to **deactivate different components** of the system – project, test plan, test case, users, build. For example, if a given test case is deactivated, it will not be included in the test plan and can therefore not be executed.

The system provides a separate module for **managing customers' requirements**. Each customer requirement can be associated to one or several test scenarios. At the end of testing, QA engineers will have a status report for the tested requirement.

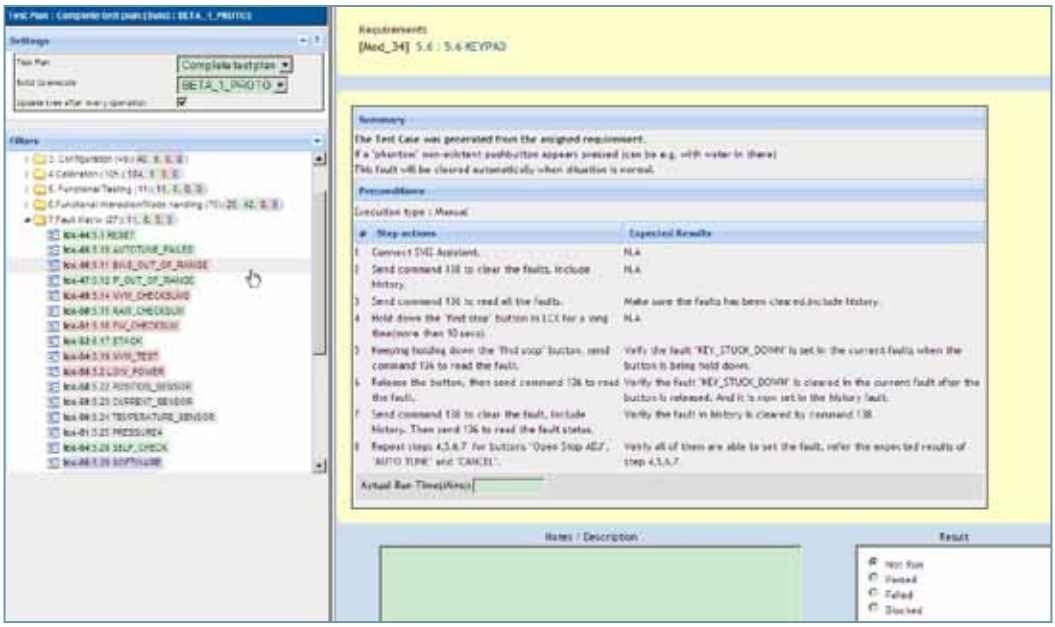


Figure 2 - Test case execution page

In TestLink you can easily **manage your test cases (scenarios)**; different versions of the test cases can be created and merged in a test suite.

Users can also copy, move, arrange and search cases in the test suite. All cases can be printed or saved in HTML and Word formats, and you could also export/import them from XML file. You can attach different files to cases, suites, and requirements.

What I also liked is that you can paste texts from Word and thus continue to follow the format of your test scenarios, which up to this moment had been created in MS Word.

By default, test cases in TestLink are assigned a status during execution, which can be pass, fail, not run or blocked.

During testing, you can generate different reports including the requirement and test case matrix report for the status of the build. The reports can be saved in MS Word and in HTML email format, which is automatically sent to the user via e-mail.

A final, but important, tip is to look into the details available in the system manual.

Good luck!

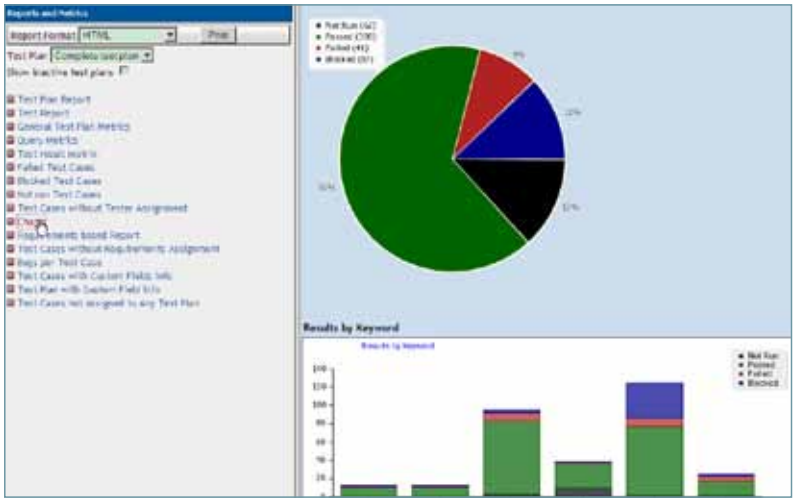


Figure 3 - Test results: Charts

Figure 4 - Test results: General test plan metrics

Biography

Terry Zuo works as Software Testing Group Leader at the Shanghai software development center, Dresser, Inc. in Shanghai, China.
He graduated from East China Normal University and is an ISTQB® Certified Test Manager. For more than 10 years he has worked in the fields of software testing and quality engineering for medical devices (*Philips Healthcare*) and process control/SMART products.
He can be reached via terry.zuo@gmail.com.

START THE NEW YEAR WITH A SMILE!

E-Exam or Paper-based*, 4 languages!*

English, German, Spanish, Russian

- ISTQB® Certified Tester, Foundation Level
- ISTQB® Certified Tester, Advanced Level
- ISEB® Intermediate Certificate in Software Testing
- iSQI® Certified Professional for Project Management
- V-Modell® XT Pro
- iSAQB® Certified Professional for Software Architecture
- IREB® Certified Professional for Requirements Engineering
- iNTACS™ certified ISO/IEC 15504 Provisional Assessor
- iNTACS™ certified ISO/IEC 15504 Competent Assessor
- TTCN-3® Certificate
- ECQA® Certified Innovation Manager
- iNTCCM® Certified Professional for Configuration Management
- iSECMA® International Certified Professional for IT-Security Management
- ISSECO® International Certified Professional for Secure Software Engineering
- RFID Management
- QAMP® Quality Assurance Management Professional

The first 200 participants who register for any available iSQI exam in January 2011 will receive an early bird discount of 11 %.

Contact our team:

certification@isqi.org
(subject line: "2011")
or via +49 331 231810-24.

*please check availability

www.isqi.org
CERTIFYING PEOPLE

iSQI®
International Software Quality Institute

Release Upgrades for Database-Driven Applications: A Quality Assurance Perspective

by Klaus Haller

1. Motivation

The majority of today's business applications are database-driven applications (DBAPs). DBAPs rely on databases for storing and processing data. When a vendor plans a new release, business analysts, software engineers, and testers collaborate to incorporate new features and to solve bugs. "The more, the better" is a typical slogan. Thus projects often postpone one task for as long as possible: the development and testing of upgrades. Upgrades, however, are of crucial importance. Customers expect a smooth transition to the new release. They do not want to lose any data. This demands for high quality assurance standards. In practice, however, not only developers but also testers treat upgrades as unloved appendices. One reason might be that the literature does not provide guidelines. This article fills this gap. It provides insights into DBAP upgrades, their risks, and adequate testing strategies.

2. Understanding DBAPs and DBAP Upgrades

The article relies on a sample DBAP, a credit risk application, to discuss challenges in a practical context. Banks use credit rating applications when they decide whether a company gets a loan and at which interest rate. Figure 1 provides a simplified model of the application. In the middle, there is the current release 8.5, release 9.0 is on the right. Shown on the left is a generic DBAP model. Release 8.5 has two input forms for financial statements. One form is for small and medium enterprises (SMEs), one for large corporations. The forms provide input fields such as "assets and liabilities" and "EBIT". The business logic calculates the company's credit risk, i.e., the likelihood that the company might not pay back the loan, based on the financial statement.

The graphical user interface (GUI) of the application with the input forms and the business logic form together the DBAP **application logic** component (Figure 1, left). It runs typically in an application/web server (**classic application logic**). Certain parts, e.g. triggers or stored procedures, are in the database (**database application logic**). DBAPs rely on three components for storing data persistently in the database. First, there is the **table structure**. It defines the data model on the database layer. The tables are a kind of container for storing customer and vendor data. **Customer data** is data relevant only for one customer. The financial statements in tables T_FINSTATEMENT_CORP and T_FINSTATE-

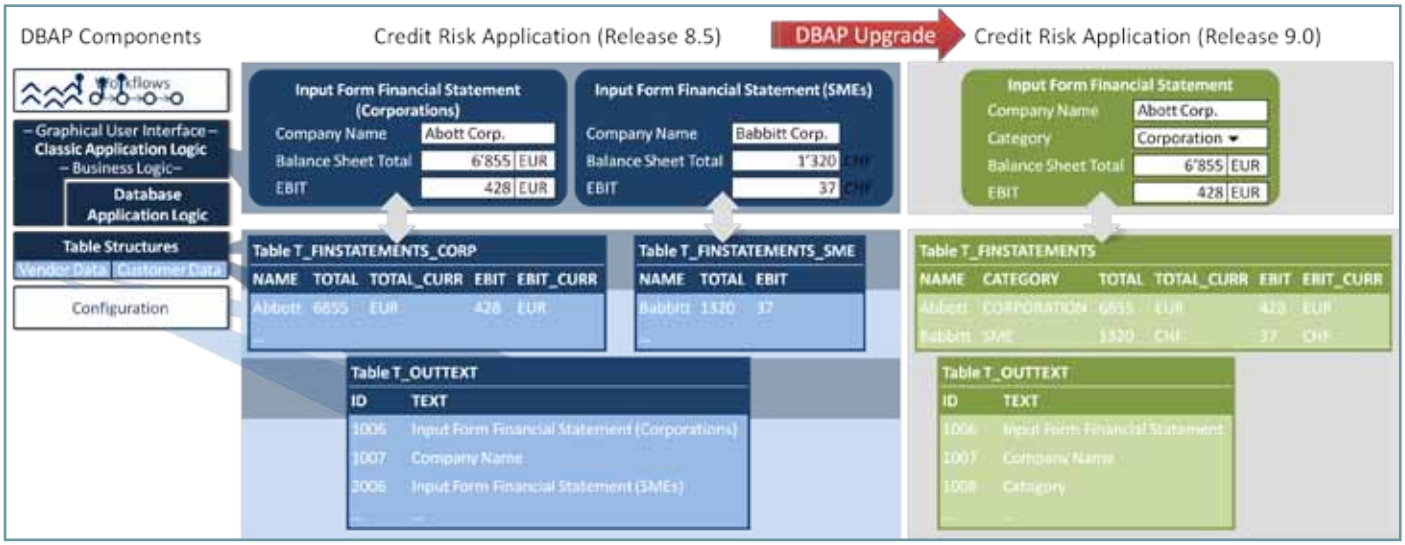


Figure 1: DBAP components (left) and sample application (release 8.5 - middle - and release 9.0 - right) with GUI (upper part) and financial statement input forms (middle and right, upper part) and corresponding database tables (lower part).

MENT_SME are examples. Table T_OUTTEXT stores the GUI names and descriptions for the input fields. These texts are shown on the GUIs. They are the same for all customer installations. This illustrates the idea of **vendor data**.

Besides release 8.5, Figure 1 incorporates also release 9.0. A **DBAP upgrade** allows switching to the new release, e.g. from release 8.5 to release 9.0. Release upgrades must overcome the differences between the two versions. Release 8.5 has two input forms, one for SMEs and one for large corporations. Both forms store their data in specific tables. In release 9.0, there is only one input form, and one table stores all financial statements. The application logic changes, too. Also, attributes, tables, and database schemas can be added, removed, or modified in a new version. A DBAP upgrade ensures that the customer can still use his old data with the new release.

3. Upgrade Correctness

Testing compares observed behavior with behavior defined to be correct. In "normal" application testing, the tester uses the applications and, thereby, enters data into input forms. If the application does not crash and the tester sees the expected output, the test is successful. Upgrades, and especially DBAP upgrades, are different. The upgrade routine must not crash. It should also return whether the upgrade succeeds. However, this does not mean that the upgrade is correct. An upgrade is only correct, if (a) the application works correctly after the upgrade, and (b) the old data is preserved and can still be used. This is an intuitive understanding, but testers need a precise understanding in order to test. The term *upgrade correctness* covers data and application component(s) aspects. Regarding most of the components, upgrade correctness bases on a comparison. It compares the upgraded DBAP with a **reference installation**. A reference installation is an installation of the new release (release 9.0 in our example). It is an installation as installed by new customers not having run any old version.

In more detail, **upgrade correctness** has four aspects (Figure 2):

1. The *application logic* of the upgraded DBAP equals the one of the reference installation. This must hold true for database application logic (triggers, stored procedures etc.) and the classic application logic, e.g., in Java/J2EE.
2. The *database tables* (the data model of the persistent data) are the same as for a reference installation.
3. The *vendor data* equals the data of a reference installation.
4. The *meaning of the customer data* is unchanged. In our example, all financial statements are still there. Also, their semantics must remain unchanged.

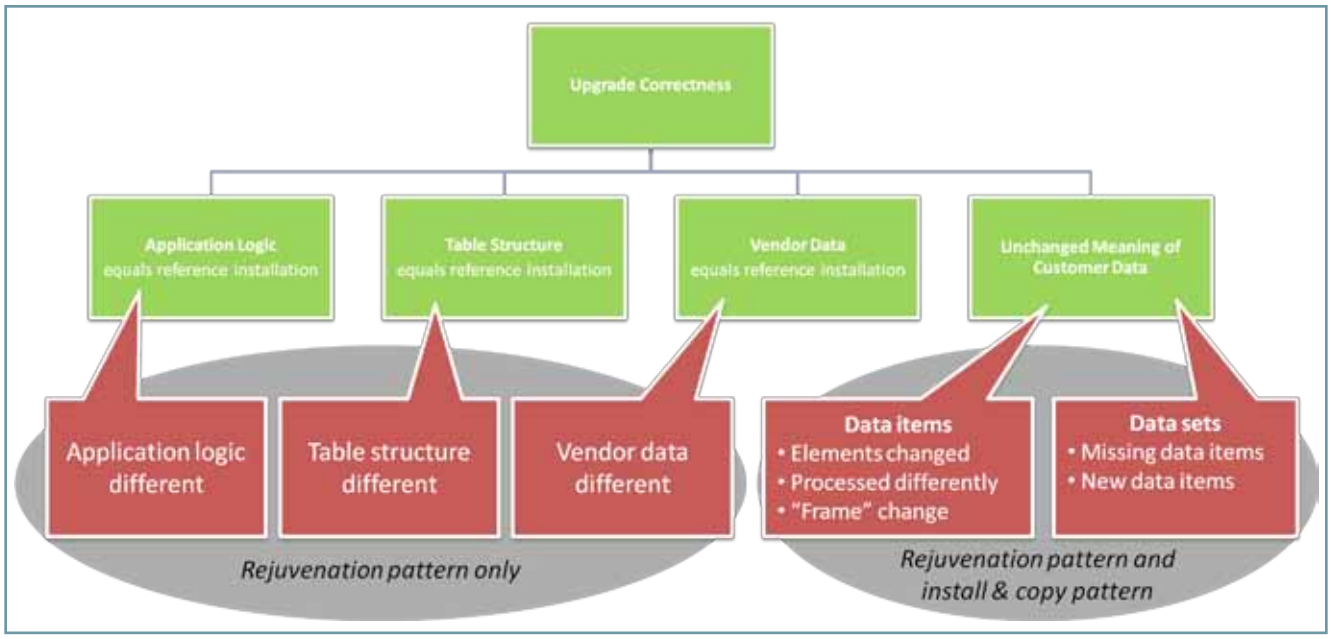


Figure 2: Upgrade correctness and upgrade risks

4. Upgrade Patterns

Software developers can use various patterns for implementing upgrades (see [1] for more details and a theoretical foundation). They have different risks. Testers need to understand the patterns and their risks for focused tests. This section presents the two main upgrade implementation patterns, the rejuvenation pattern and the install & copy pattern.

The **install & copy pattern** installs the new DBAP version in step 1 (Figure 2, 1). This is done the same way as for new customers. In step 2, the upgrade copies the data from the old installation. It transforms the data and loads it into the new installation (2). As a result, there are now two installations. One is the old one (release 8.5 in our example). It can be shut down and removed from the server in step 4. Before that, in step 3, the users switch to the second, the new installation. The only upgrade risk of this pattern is that the meaning of the customer data might be changed. All other components (application logic, vendor data, and table structure) are installed from scratch. Thus, they are correct by definition.

The **rejuvenation pattern** adopts the application logic. It removes obsolete and deploys new Java classes (Figure 2, A). It deletes, modifies, or adds database application logic such as triggers and stored procedures (B). It modifies database tables and adopts customer data accordingly (C). Finally, it updates the vendor data (D). So the rejuvenation pattern transforms the old installation into a new one. The pattern has various risks. First, the meaning of the customer data might change. This risk is the same as for the previous pattern. Other risks are that the application logic, the table

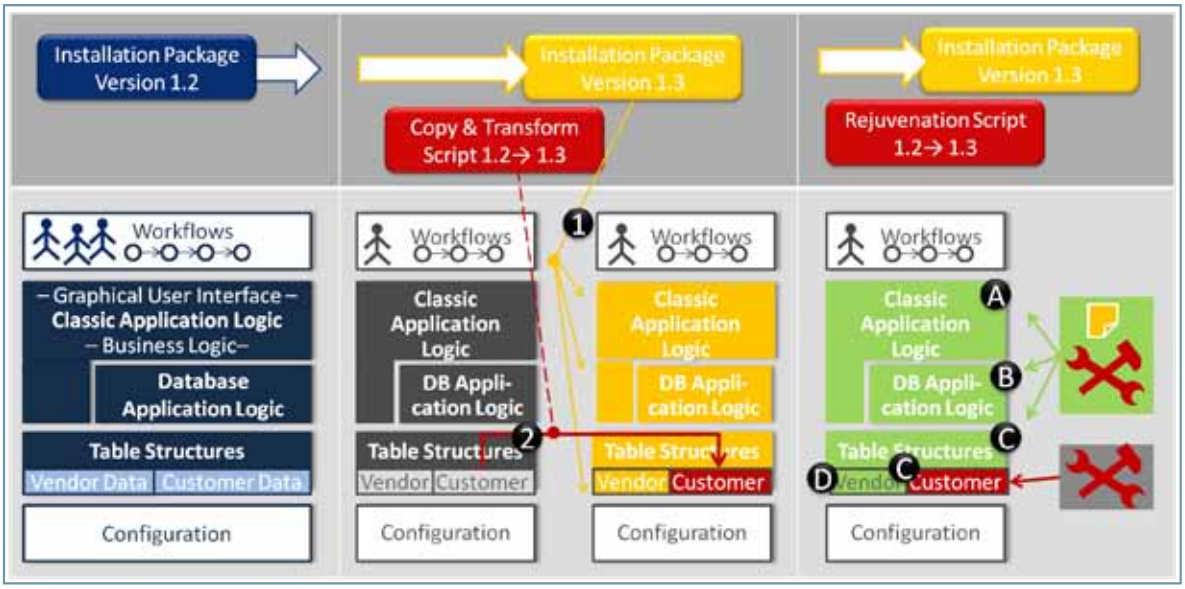


Figure 3: Upgrade patterns with the old version (left), the copy & transform upgrade pattern (middle), and the rejuvenation upgrade pattern (right)

structure, or the vendor data are modified incorrectly. Then, not all components of the DBAP equal the reference installation. The DBAP is inconsistent. It might return wrong results (e.g., wrong credit ratings) or simply crash.

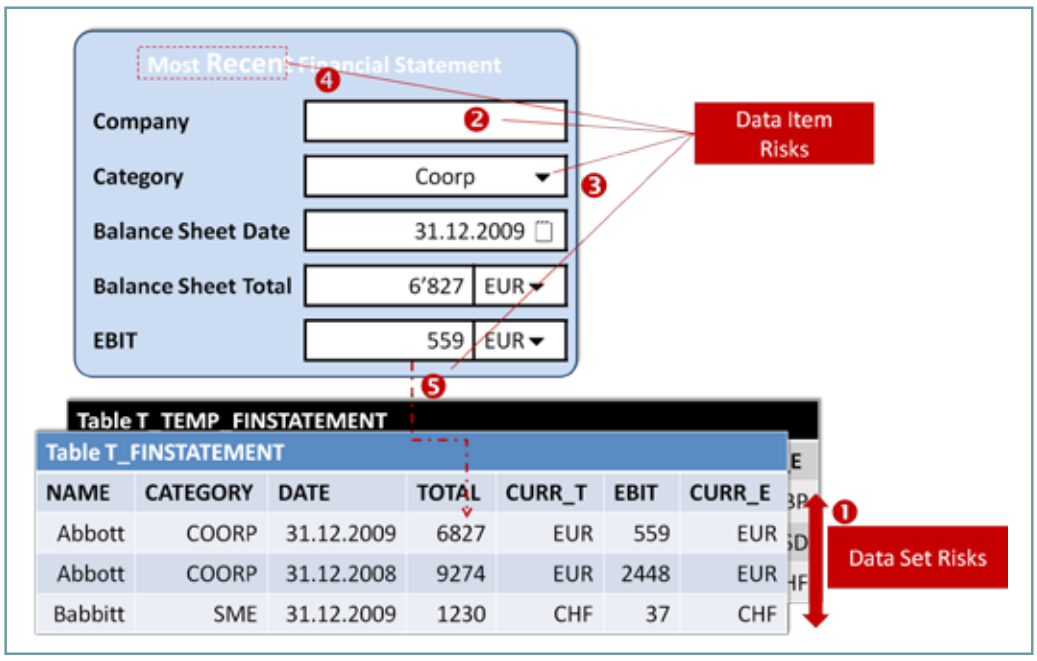
5. Understanding DBAP Upgrade Risks

The previous section mentioned briefly the upgrade risks. Three are easy to describe. Application logic, table structure, and vendor data must be the same as for a reference installation. If not, the upgrade is not correct. Customer data-related risks are more complex. They fall into two sub-categories. First, there are two **data set risks** (Figure 4, 1):

- *Missing data items*. One or more data items might get lost during the upgrade. There could be twenty financial statements in release 8.5, from which one is missing when the bank starts working with release 9.0 after the upgrade.
- *New data items*. The upgrade must not "create" financial statements which did not exist before the upgrade. This could happen, e.g. when certain financial statements are duplicated unintentionally.

The **data item risks** deal with the semantics of a single data item, e.g. one financial statement. There are three risk types:

- *Element changed*. One or more attributes are changed incorrectly. Attributes might get lost during extraction and copy. If the upgrade uses an INSERT-SELECT-SQL statement, it might not cover all needed attributes (2). This seems unlikely for our example with table T_FINSTATEMENTS_CORP or T_FINSTATEMENTS_SME. However, if there are one hundred or more tables, forgetting one attribute is not so unlikely any more. Also, the table structure of the new release can change till the last second. This requires changing the upgrade always accordingly. Regarding incorrect changes, an example would be if the currency for all SME financial statements is not set to CHF during the upgrade, but to USD.
- *Processed differently*. The category field is a good example how an additional attribute influences the processing of otherwise unchanged data (3). If one copies old financial statements for release 8.5 and does not fill up the new attribute CATEGORY, the business logic might use the wrong algorithm and calculate wrong credit ratings, or it might crash.
- *Frame change*. The data is the same on the database level and is (still) correct. However, the semantics can change due to (slightly) different GUI texts ("most recent financial statement" vs. "last year's financial statement" – 4), due to field name changes (e.g., "total" instead of "EBIT"), or due to being linked to a new GUI or differently to the existing one (5).



6. Testing in Practice

Vendors and customers test for different reasons. Vendors want to improve the quality of the upgrade. It must run with all customer installations, even if various installations differ sub-

Figure 4: Interpretation-related risks: data item risks and data set risks

tly. Customer-site tests ensure that the customer company can continue working. Companies cannot risk that an ERP system is down for a day or two. Banks cannot approve loans without credit rating. So upgrades are done in maintenance windows. The application management tests whether the company can *work* with the upgraded DBAP. If not, it restores the state before the upgrade. Thus, the business is not affected even if the upgrade fails. The good news is that the customer and vendor tests might have different objectives, but the needed techniques overlap.

The DBAP upgrade risks are the base for deriving the techniques and strategy for testing DBAP upgrades. They guide the testers when preparing test plans and running tests. Table 1 compiles a DBAP upgrade test strategy. It links the risks to the tasks to be performed. It states also whether automation is an option for vendor-site or customer-site tests.

Table 1: DBAP Upgrade Test Strategy

Patterns	Risk Type	Risk Subtype	Test Approach	Automation (Vendor-site)	Automation (Customer-site)
Rejuvenation only	*Application Logic Differences*	"Classic"	*As for normal upgrades*		
		Database	Schema comparison	Yes, full automation possible	Yes, full automation possible
	Table Structure Differences		Schema comparison		
	Vendor Data Differences		Counting rows/ calculating hash values		
Install & Copy and Rejuvenation	*Data Set Risks*	*Missing Data Items*	Reconciliation		
		New Data Items	Reconciliation		
	Data Item Risks	*Element change*	Checking all representatives	Questionable cost-benefit ratio	No (tests only needed once)
		Frame change	Checking all representatives	No	*Irrelevant*
		Processed differently	Execute *all* workflows with as much data as	Questionable cost-benefit ratio, option: smoke	Part of "normal" tests after setting up a system

There are the three risks specifically relating to the rejuvenation pattern: application logic differences, table structure differences, and vendor data differences. The database catalog helps addressing two: **database application logic differences** and **table structure differences**. It contains a list of all existing tables and their attributes. The vendor extracts this information from a reference installation and incorporates it into the upgrade. At customer-sites, when the upgrade has finished with all transformations, two validations take place. The first validation compares (i) the (database) application logic and table structure after the upgrade with (ii) a reference installation. It queries the database catalog of the DBAP for (i) after the upgrade. The information for (ii) is incorporated in the upgrade. A second validation checks **vendor data difference** risks. It counts, e.g. the rows per table and calculates hash values. The vendor benefits from the validations in two ways. First, developers get quick feedback during the upgrade development. Second, if run at customer-site, the vendor knows that the upgrade ran as expected. If the vendor has not considered all nuances of a customer installation, they are either detected by the checks, or they are irrelevant. If upgrade problems are not detected early, this can ruin the vendor reputation. Without remote access to the customer environment, the vendor support has no chance identifying bugs which are a result of such upgrade "hiccups".

Testing **data set risks** means matching data items of the old and new system. Data items without matching counterparties indicate errors. It is like looking for a (missing or an dditional) needle in a haystack. It is not about whether the needle head is blue or red. The checks do not consider any semantics. So automation is an option. A reconciliation is the technical solution. A *reconciliation* is a list comparing, per table, identifiers before and after the upgrade as Table 2 illustrates. The top four rows contain rows about loans. They are identified by their IDs. There are no matches for account 200540. It exists only in the new version. Contrary, account ID 755543 exists only before the upgrade.

In practice, a reconciliation must consider three advanced aspects:

- Certain data items are not relevant for upgrades, e.g. logs. Also, if modules or functionality is removed, corresponding data becomes obsolete. They are not looked at by the reconciliation.
- Changes of the table structure must be addressed. When the data of the two tables T_FINSTATEMENT_CORP and T_FINSTATEMENT_SME is copied into one table T_FINSTATEMENTS, the reconciliation must consider this. The artificial table name "OBJ_FINST" reflects this. It is used for elements in all three tables.
- If a table has no identifier, a good one can often be constructed by concatenating attributes, e.g. a company name and the balance sheet total in the example.

The data set risks are so crucial that vendors must address them during their tests. In case of business-critical applications, customers (or their auditors) might insist on such checks when the upgrade takes place at customer-site. Thus, the vendor should design a reconciliation as delivery object for the customer and run them after/with each upgrade. The concept of a reconciliation is discussed in more detail in [2], e.g., on how to integrate more attributes.

Risks which involve semantics are difficult to automate. They depend on human testers. This applies to two data item risks, the element change risk and the frame change risk. They are done at customer-site and vendor-site. Obviously, it is not possible to check every data item, but only a small selection. Thus, it is important to choose representatives. Testers must have a list of the ty-

Table 2: Reconciliation

Table T_FINSTATEMENTS			
TAB_OLD	ID_OLD	TAB_NEW	ID_NEW
		T_LOAN	200540
T_LOAN	422784	T_LOAN	422784
T_LOAN	522423	T_LOAN	522423
T_LOAN	755543		
OBJ_FINST	Abbott6855	OBJ_FINST	Abbott6855
OBJ_FINST	Babbitt1320	OBJ_FINST	Babbitt1320

pes of data items in the system, e.g. financial statements, customers, loans, etc. If there are subtypes with different behavior or processing, e.g. financial statement of a SME company versus financial statements of large corporation, this must be considered. Then, testers choose representatives for each data item type. The more complete and varied the set of representatives, the likelier it is that failures are detected. Practical experience proves, that all attributes of all input and output forms must be covered at least once.

Testers compare the data before and after the upgrade manually. They print out the relevant data before the upgrade and compare the print-outs with the data in the upgraded DBAP. In case of the install & copy pattern, testers can also log into the old and the new systems simultaneously. Then, they compare the data items by opening the relevant forms in both systems. This takes time, i.e. it is an expensive task. Especially for the upgrade development project, recurrent element change tests might be automated. This requires test data in the old installation, which is tested at GUI level after the DBAP upgrade. There is no such option for frame change tests.

The processed differently risk is trivial when one is aware of it. However, at the very end of a project, a tester had compared all representatives thoroughly. There was not a single error left when looking at the data in the GUI. By chance, one developer pushed a button starting some data processing of/with the upgraded DBAP. The whole application crashed immediately. The reason was a new mandatory attribute in a table. The GUI presented a default value if the attribute was missing. The application logic crashed. This illustrates that all workflows should be executed respectively tested at least once by a tester manually. Then, the tester can also check whether the returned results are as expected. This approach turned out to be highly effective. In case of longer upgrade projects, implementing smoke-tests for all important workflows can speed up the development process.

***

Developing and testing upgrades are crucial for customer satisfaction, especially for DBAPs. Customers expect, first, that the application runs as smoothly as before the upgrade. Second, all data must (still) be available. This is challenging. After reading this article, testers should understand three aspects better. These are, first, the DBAP upgrade patterns, second, the risk associated with the upgrade patterns (precisely: data set risks and data item risks), and, third, what kind of tests are needed. This knowledge makes testing DBAP upgrades more focused, more effective, and, hopefully, more fun.

References

[1] K. Haller: *On the Implementation and Correctness of Information System Upgrades*, International Conference on Software Maintenance (ICSM), 12-18 September, Timișoara, Romania

[2] K. Haller: *Data Migration Project Management and Standard Software*, Data Warehousing 2008: St. Gallen, Switzerland

Biography

Klaus Haller studied computer science at the Technical University of Kaiserslautern, Germany. He received his Ph.D. in databases from the Swiss Federal Institute of Technology (ETH) in Zurich, Switzerland. Klaus has worked now for more than five years mainly in the financial industry. His focus is on information system architecture & management and business-IT alignment.

Why we afraid of open-source testing tools?

by Lilia Gorbachik

When you start the new project, one of the activities is to select the list of testing tools. Some big companies have a standard that defines tools, but we are interested in a different case. So, you have to face to the world of tools. Some of them are multi-functional, some of them produced for narrow purposes, some of them open source, some of them cost a huge amount of money. So, you have the multi-dimensional space of metrics that will help you solve the issue of selecting.

At this step I often could hear from the manager: "No, we'll exclude open-source tools the list of interest". When I queried: "Why???", the answer was: "I don't want to have additional future problems with testing tools. We'll be under pressure in this project, so I want to decrease third-party software risks."

That is how software managers see the software development process. They face budget, time, human resources and motivation problems in a project. They try to compare their own project with some view of an open-source project. However, there are certainly different types of software involved that use different rules. That is the main misunderstanding.

Let's look at the myths and dispel them all.

Myth #1: Open-source tools are not as stable as other commercial off-the-shelf products

Building a team is a real art as every manager would confirm. So how can independent people that are occasionally involved in an open-source project produce high quality product and work as a team? This is the cause of doubts about open-source products. However, one must note that people are motivated enough to enter the project. Those who decide to participate in an open-source project feel it's their own responsibility and influence the project. It is just fun and a sort of challenge. They try to do their best because it is a collaborative piece of work. Everybody can inspect each other's work and project members clearly understand this. So, don't worry about low quality.

Myth #2: There is no ability to train people before start.

Come on, be honest. We don't train testers before starting to build every new testing framework. This is reality. Testers often learn via articles, podcasts, forums etc. Yes, it's true. So, what is the difference? If you use open-source testing tools, you usually even have the opportunity to ask the author about the testing tool, about particular features or suggest new features. You could get closer contact with project members. This is a benefit, so don't forget to use it!

Myth #3. Open-source testing tools could be cancelled because of money.

This may happen with every project that doesn't meet customers' expectations. The advantage of open-source testing tools is that there is a close connection to the end users. Forums, boards and similar ways of communicating all reflect the expectations and the possible development of the tool. There are a lot of open-source products that are successful. Everybody knows open-source products like Linux and Open Office , but nobody wants to debate here about the survival of open source products.

Myth #4. If we find a bug, it will be hard to get a patch.

Oh, it's again about the quality... But, do your customers get the patch immediately? No. So, it's real life. To get a patch for the issue is not easier but also not harder than for any commercial product. So, relax.

And the final argument; talk to your developers. I'm convinced that they use a lot of open-source software for personal and work purposes. Maybe developers are the most active evaluators of open-source products. Share with them your doubts and you will see how powerful open-source tools could be. We all have limited budgets for teaching people the use of testing tools. So, by decreasing the testing tool costs you could increase the training budget for your staff; make them more efficient, more powerful, more flexible.

So, to achieve product quality, we permanently search for improvements for the testing and developing process and try new approaches, new methodologies. Now is the time to try a new type of the testing software; open-source tools. In some cases, you may have heard about the tools, but just didn't realize that they are actually open source. Refresh your mind.

Another advantage; you could customize the testing tools for your project. This is more important if your project is specific in some way and it's hard to find solutions, or when you need just a particular feature and don't need hundreds of features included in products for a wide spectrum of use.

Just try it once and I'm sure you'll not be disappointed.

Lilia Gorbachik has enjoyed working in software development. She has experience in product and outsourcing companies. Lilia contributes articles about testing to Russian testing communities. She has participated in SQA Days conference that is the most important testing conference in Russia.

Orthogonal Approach To Reduce Test Effort Via An Open-Source Tool

by Saurabh Chharia

Reduction of testing effort without compromising on the quality is one of the prime concerns for all software applications. During testing of an application, at least 40-45% of testing effort goes into test case preparation. An increase in testing effort for manual testing can highly impact not only the productivity of the vendor, but also the budget of the client hence it has a twofold effect. This article discusses the issues, challenges and solution to enhance the productivity and quality of the deliverable. It explains the differences to a conventional approach for creating test cases. It presents an approach to optimize test case creation effort without affecting test coverage and quality of the deliverable.

(WHY)?

Testing effort can be drastically decreased by identifying the pair-wise combination for the input conditions during the test set data creation phase. This can be done using a concept of Orthogonal Array which can be implemented using the open-source tool called 'Allpairs'. This open-source tool reduces the input conditions required to test a workflow without compromising on the business rules coverage of the application.

This article presents an approach for reducing the test effort by using the open-source tool "Allpairs" which is based on the Orthogonal Array concept.

Problem Statement: (WHAT)?

In a traditional test case preparation approach, we write test cases for all the permutations and combinations of inputs, which is why the testing effort increases. This not only brings down productivity during the test case creation phase, but also increases costs.

The statistical approach to implement the Orthogonal Array concept using the open-source tool Allpairs: - (HOW)?

Solution:

- Identify the business parameters/factors/functionalities on which the requirement document focuses.
- Identify possible data values of each parameter/factor.
- Install the Allpairs software, then unzip the Allpair files in the local machine.
- Quantify the parameters and values for the functionality to be tested in a tabular format and save the file as a text document (the contents of this file should be tab delimited).
- Open the command prompt and execute the file with-
- ALLPAIRS Input.TXT > Output.TXT
    - Where Input.txt - The file which contains the parameters and values.
    - Output.txt - The file which stores the output.

"Input file for Allpairs"

Why is Pair-Wise testing so proficient?

Example: Payment–Gateway UI application.

- Suppose a bug exists for the input conditions as **First name**: Greater than 12 characters and **Bank Name**: AXIS radio button are selected.
- The defect can be found using the pair-wise concept.

Card Payment (CheckBox)	Bank Name (Radio Buttons)	First Name (Text Box)
Cheque	ICICI	Saurabh
Credit Card	AXIS	Null
Debit Card	HDFC	Greater than 12 characters

"Possible combinations"

Using Exhaustive testing

Example: 3*3*3 = 27 tests cases.

Cheque	ICICI	Saurabh
Cheque	ICICI	Null
Cheque	ICICI	Greater than 12 characters
Cheque	AXIS	Saurabh
Cheque	AXIS	Null
Cheque	AXIS	Greater than 12 characters
Cheque	HDFC	Saurabh
Cheque	HDFC	Null

Cheque	HDFC	Greater than 12 characters
Credit Card	ICICI	Saurabh
Credit Card	ICICI	Null
Credit Card	ICICI	Greater than 12 characters
Credit Card	AXIS	Saurabh
Credit Card	AXIS	Null

"Defect input condition detected"

Pair-Wise Results

9 combinations are detected using the orthogonal concept:

Test#	Card Payment (Checkbox)	Bank Name (Radio Buttons)	First Name (Text Box)
Test1	Cheque	ICICI	Saurabh
Test2	Cheque	AXIS	Greater than 12 characters
Test3	Cheque	HDFC	Null
Test4	Credit Card	ICICI	Greater than 12 characters
Test5	Cheque	AXIS	Null
Test6	Cheque	AXIS	Greater than 12 characters
Test7	Cheque	HDFC	Saurabh
Test8	Cheque	HDFC	Null
Test9	Cheque	HDFC	Greater than 12 characters

Conclusion

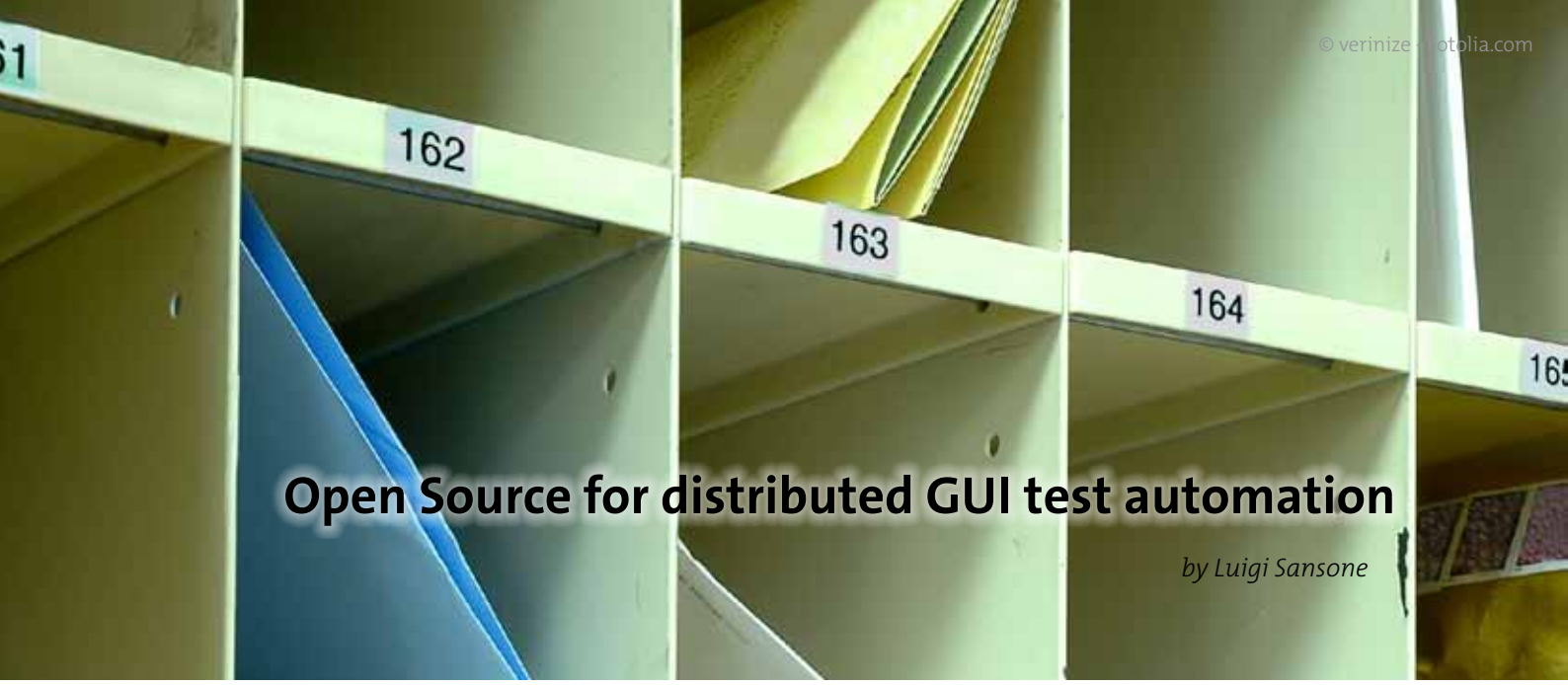
An orthogonal approach to optimize test case creation and test data set not only, works very well for requirement capture, effort estimation, test data and test case planning but also for testing different modules across different components within the same application. This approach for test case creation proves to be very useful in capturing the best input conditions, which cannot be captured by traditional means. The orthogonal approach can provide the most effective mechanism for test case and test data designing.

Disclaimer

The information contained in this document represents the views of the author(s) and the company is not liable to any party for any direct/indirect consequential damages.

Biography

Saurabh Chharia is an experienced automation and manual testing professional who works for HCL Technologies. He is an engineering graduate who has a fanatical interest in writing articles, papers and artifacts in the software testing field. One of his articles on test automation was published in 'Testing Experience'. He has practical knowledge of domains such as retail, insurance, reinsurance and banking.



Open Source for distributed GUI test automation

by Luigi Sansone

This article describes a case study on the validation of rich UI-based applications through a distributed and open-source test automation framework. The article presents an environment consisting of several independent, customizable and loosely coupled components (UI viewer, test object finder, robot ...) for functional and regression testing of Java applications. Results, advantages and issues will be presented.

Even though giant steps have been made during the last decade towards a more effective test automation, the most used techniques and tools are still lacking in terms of flexibility being too application dependant. This can mean that any change to the system under test, for example to the user interface (UI), requires considerable time–consuming refactoring of the produced test scripts, thus precluding the possibility of a widespread reuse.

GUI Maps

Graphical User Interfaces (GUIs) are everywhere: computers, hand-held devices, household appliances and office equipment.

In our experience, a lot of time is spent on creating and running automated functional GUI tests to produce more reliable and usable software systems.

One of the major challenges in test automation is to deal with GUI controls. Many automated testing tools use the so–called 'GUI maps' – a.k.a. 'Test Object Maps', namely repositories of the objects relevant to the application under test (AUT) - to manage the graphical elements in the system.

The map is a static view that describes the test objects known to the automation tool, so that the test scripts may refer to the recognition properties contained in it to retrieve any object. When the automation tool runs a test, it uses the GUI map to locate the objects to test. The tool reads an object's description in the map and then looks for a GUI object with the same properties in the application being tested.

The GUI maps for an application are created while test cases are being recorded – in the "capture and reply" mechanism – or by using a spy tool.

GUI Maps are a pre-requisite for creating and running the automated tests. For this reason, testers need to wait for an early release of the application to be tested in order to capture the map. This makes the development of automated tests before the final AUT availability virtually impossible. By this approach test automation will always start later in the software lifecycle, stretching the testing time.

Moreover, just as any software artifact, GUI maps need to be maintained: in case of changes, testers have to update or (in most cases!) recapture the maps to reflect changes. And GUIs usually change very often!

Maveryx: Beyond GUI maps

To address these typical problems, we designed Maveryx[1], a new, Java, open- source test automation tool for functional and regression testing of Java GUI-based applications.

In contrast to other tools, Maveryx scans the user interface and identifies the GUI objects to be tested at runtime during script execution. No GUI map is required, just code and run.

This is made possible by a *Viewer* (Figure 3) component that captures automatically, in a standard XML representation called 'snapshot', the hierarchical structure of the current application's UI.

Each snapshot is processed by an intelligent *GUI Objects Finder* (Figure 3) that recognizes and locates the test objects using some advanced, fast and efficient searching algorithms.

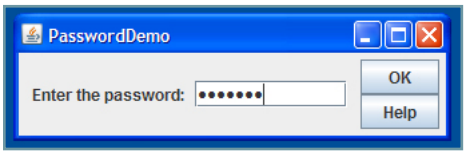


Figure 1 - An example of GUI

For example, using the provided Java API, if the test script contains the following instructions (i.e. click the 'OK' button in the application's frame 'PasswordDemo' – Figure 1) ...

1 *http://sourceforge.net/projects/maveryx/*

Probador Certificado Nivel Básico

Tester profesional de Software

Formación para el Probador Certificado - Nivel Básico
de acuerdo al programa de estudios del ISTQBâ

República Argentina

Docente: Sergio Emanuel Cusmai

Co - Founder and QA Manager en QAustral S.A.
Gte. Gral. de Nimbuzz Argentina S.A.
Docente en diplomatura de Testing de Software de UTN - 2009.
Titular y Creador de la Diplomatura en Testing de Software de la UES XXI - 2007 y 2008.
(Primer diplomatura de testing avalada por el ministerio de Educación de Argentina).
Team Leader en Lastminute.com de Reino Unido en 2004/2006.
Premio a la mejor performance en Lastminute.com 2004.
Foundation Certificate in Software Testing by BCS - ISTQB. London – UK.
Nasper - Harvard Business school. Delhi – India.

Casa Central: + 54 0351 4817951 - **Tel Bs As Argentina:** 011 53685847 - **Tel Londres UK:** 00442033183814
Tel San Francisco. EEUU: 001 14157041515 - **E-mail: info@qaustral.com**

```
GuiFrame f = new GuiFrame("PasswordDemo");
GuiButton ok = new GuiButton("OK", f);
        ok.click();
```

... will trigger in Maveryx, at runtime and transparently to the user (Figure 2), the following process:

1. the Viewer catches the snapshot of the current application's UI
2. the GUI Objects Finder processes the snapshot searching for a GUI object of type='button' with caption='OK'
3. when found, the button reference is passed to the robot for pushing.

Using this approach, testers don't need to provide any apriori GUI map containing the 'OK' button.

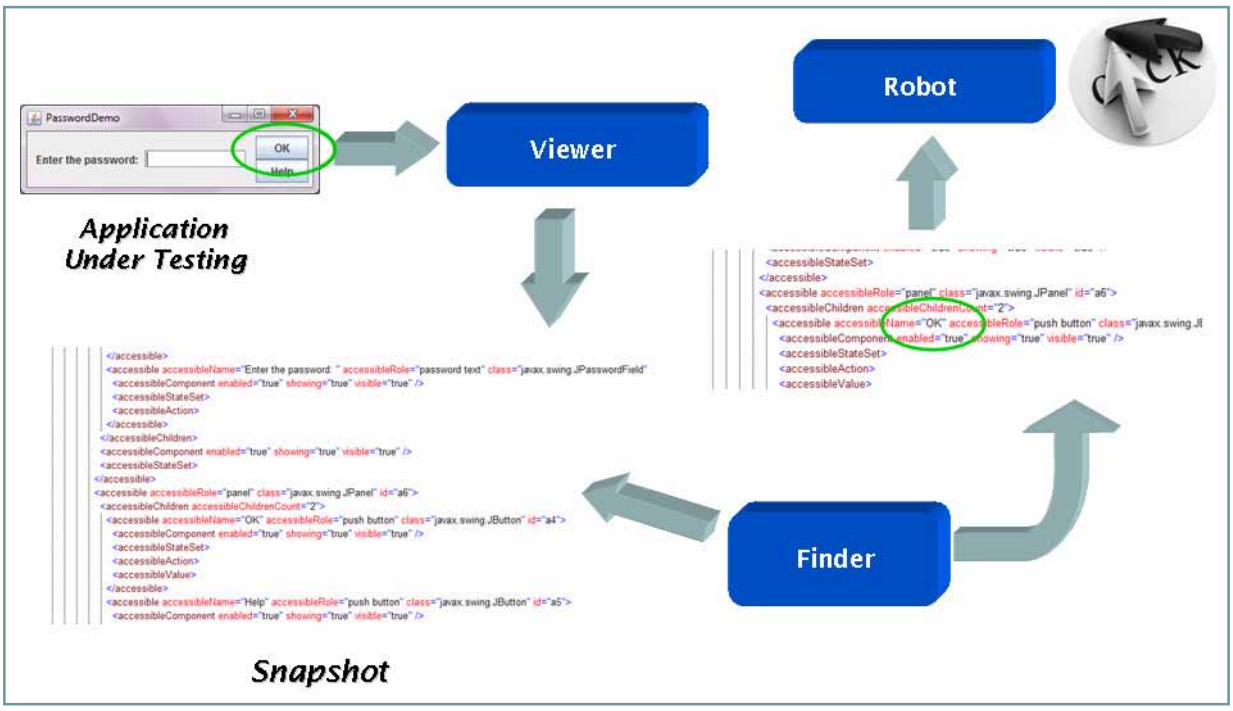


Figure 2 - Maveryx main working flow

GUI Object Finder

Test objects defined in the scripts are identified and located directly at runtime by an advanced *GUI Objects Finder* with several pattern–matching capabilities.

This is a lightweight XML-based search engine powered by a wide range of criteria, from classical to fuzzy.

The search criteria provided allow to uniquely identify the GUI object to test in case of 'exact', 'partial' or 'approximate' matching (Table 1).

Criterion	Description	Example[2]

Exact Match	Allows searching for test object exactly as it is entered	`GuiFrame f;`
		`f = new GuiFrame("PasswordDemo");`
		`f.close();`

Case Insensitive Match	Allows finding an exact match, but the match is not case-sensitive	`GuiFrame f;`
		`f = new GuiFrame("passworddemo");`
		`f.close();`

Partial Match	Allows finding a match on partial data	`GuiFrame f;`
		`f = new GuiFrame("Demo");`
		`f.close();`

Wildcard	Allows finding a match by using regular expressions	`GuiFrame f;`
		`f = new GuiFrame("P?w?Demo");`
		`f.close();`

Levenshtein Distance	Allows matching approximate strings with fuzzy logic, using the Levenshtein distance	`GuiFrame f;`
		`f = new GuiFrame("PWDDemo");`
		`f.close();`

Thesaurus Searching	Search for broader or narrower or related terms or concepts in a vocabulary	`GuiButton ok = new GuiButton("O.K.");` `ok.click(); // O.K. -> synonym of OK`
Google Translation	Allows finding an exact match translation	`GuiButton ok = new GuiButton("Aiuto");` `ok.click(); // Aiuto -> English Help`

Table 1 - Maveryx searching criteria

[2] See Figure 1

The *Levenshtein Distance* allows finding approximate or fuzzy matches in terms of edit operations (insertion, deletion, or substitution of characters).

The *Thesaurus Searching* allows finding GUI objects using synonyms, from the WordNet[2] dictionary.

The *Google Translation* criterion is designed for internationalization testing and low-level components linked to the operating system language.

Moreover, changing the user interface, for example adding something, will often break the test cases which depend on the modified parts. Typically, even minor changes to the GUI layout break the test scripts. GUI modifications make a large number of test cases unusable, requiring expensive update.

The Maveryx search engine allows searching and locating the GUI controls to test, even if the objects have changed since test creation or 'partial' descriptions are given.

The tester can configure the object–matching sensitivity to run the test scripts successfully, without changing the scripts, even when the AUT has been updated. Fuzzy algorithms combined with a robust test design, allow to automate tests resilient to frequent application user interface changes, avoiding continuous reworking.

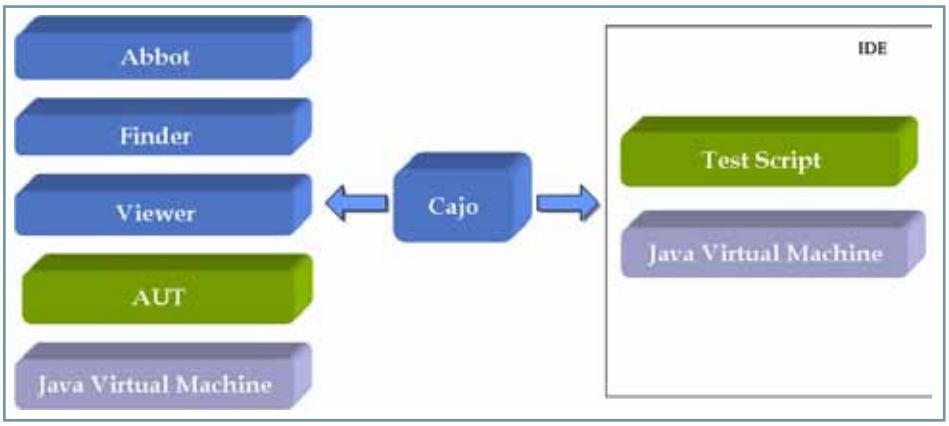


Figure 3 - Maveryx architecture

Open is better

Maveryx is open source. This choice to develop Maveryx gave (and will give) us the possibility of maximizing the reuse of the best-in-class open technologies and software (Abbot[3] for the robot, Cajo[4] for communicating between JVMs ...) we need, without reinventing the wheel.

The main benefit for users is the possibility to take all the advantages of open-source software, first of all, the availability of the source code and the right to modify it for unlimited tuning and improvement. Open also means 'extensible'...

Maveryx has a plug-in system to customize and extend the core platform for the specific testing needs.

Plug-ins allow extending the Viewer engine for recognizing custom controls as well as to support new platforms other than Java.

The GUI Object Finder can be also extended with new search criteria to best fit the test objects in a specific application domain (e.g. telecommunications, banking, etc.).

Maveryx provides also a comprehensive set of high-level Java APIs for programming the test scripts. Users can create their own API to extend Maveryx with new testing capabilities.

Results and benefits

We have used Maveryx in several test projects belonging to different application domains (telecommunications, health systems, IDE ...), both in traditional and in Agile environments. We combined the tool features with an effective data-driven approach to separate tests from UI data, to avoid continuous changes to the scripts. This approach gave us the possibility of creating test scripts resilient to frequent UI changes.

The main benefits we gained were:

1. starting automation early in the software lifecycle (long before the application was released for testing),
2. executing the automated tests more often,
3. gaining more time to create more tests (hence less manual testing),
4. reducing the test maintenance effort between releases, and
5. achieving a higher reuse (up to 40%) of scripts through different test projects.

In our experience, running test automation in parallel to software development, rather than in cascade, significantly reduced the overall time (up to 30%) and the cost-per-bug (up to 40%).

Replacing UI maps with an effective data-driven approach cut the maintenance effort up to 40%.

Decoupling the scripts from the GUI maps allows writing the automated tests early in the software lifecycle, for example, just after the specification phase, running them immediately after the application is released for testing.

Removing the dependency from the GUI Maps, hence from the particular application under test, allows porting of the same testware, with less effort, to enable use in different testing projects with different applications.

These encouraging results convinced us to share Maveryx with the community as an open-source project.

2 *http://wordnet.princeton.edu/*

3 *http://abbot.sourceforge.net/*

4 *https://cajo.dev.java.net/*

Biography

Luigi Sansone is one of the Maveryx Project Leaders. He has more than 10 years experience in software testing and quality assurance as tester, test consultant, and test manager. He is primarily specialized in test automation and test management. Luigi has managed a variety of test projects of different team sizes and complexities, many in the telecom sector, by testing both stand–alone and distributed applications. He teaches courses on test automation and is speaker at conferences, universities, and corporate events.

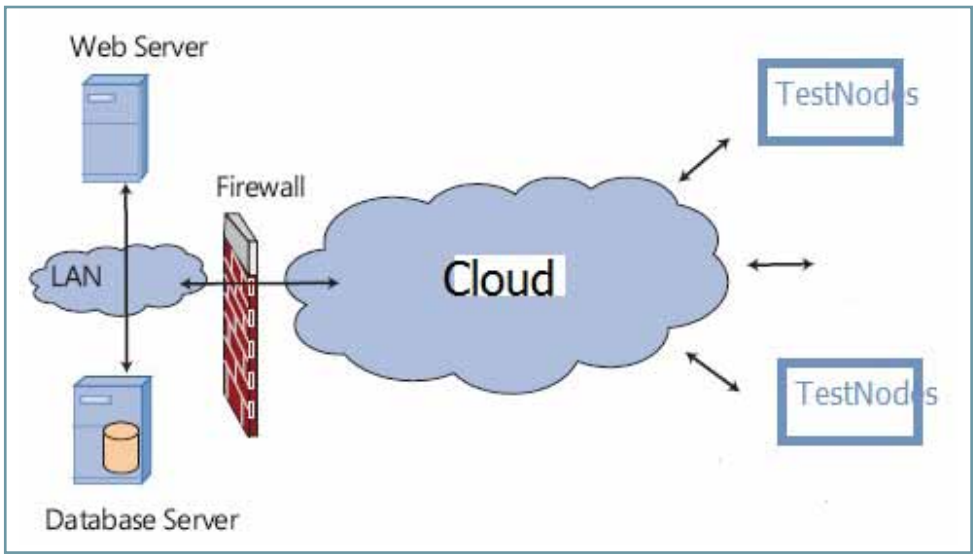
Open-Source Testing with Infrastructure-as-a-service

by Frank Cohen & Saurabh Mallik

The IT world is headed towards Cloud Computing and using infrastructure (servers, storage and network) as a service. The Cloud has brought a disruptive innovation to the world of software testing. In this article the authors discuss how open-source Cloud testing tools can leverage the benefits of Cloud infrastructure like Amazon EC2, Collabnet, Rackspace or GoGrid to scale existing performance tests and run tests which were never possible in the data center.

Are you ready to move your data, applications, and services into a brand new technology platform? Moving to the Cloud can be unnecessarily dangerous and risky. On the other hand, moving to the Cloud may be an inevitable way to power your organization's IT initiatives in Rich Internet Applications (RIA,) Service Oriented Architecture (SOA,) and Business Process Management (BPM). Test and development teams are striving to adapt business service management, load testing, and application service monitoring to be the proven techniques, tools, and methodologies to enable secure and profitable IT operations in the Cloud.

Let us start by defining what "the Cloud" really means. The National Institute of Science and Technology (NIST) defines the Cloud as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources, for example networks, servers, storage, applications, and services, that can be rapidly provisioned and released with minimal management effort or service provider interaction. The service model of the Cloud is divided into three:- Software as a service (Saas), Platform as a service (Paas) and Infrastructure as a service (Iaas). Infrastructure as a service is access to computing resources, like servers and storage, in an on-demand and scalable nature. We provision server time and network storage as we like. This is the pay-as-you-go model where users plan and buy the computing resource they want for their testing and development needs. Key popular vendors in the Iaas space are Amazon, Rackspace, GoGrid etc.



This leads us to the application of Iaas to testing.

Defining Cloud Testing

Any functional, performance or load test, and production monitor that uses Cloud resources like infrastructure, application or platform would be categorized under Cloud testing. However, we feel that for a test to truly leverage the Cloud, the following criteria need to be met:

a) Scalability & On-Demand Testing: Users must be able to vary the number of test nodes (also known as test injectors) to achieve the load. For a test to be truly scalable, the test group must be able to increase the load, sometimes up to millions of users. A second key requirement is that the management of the resources should be automated and rule-based. Cloud computing makes it much easier to provide on-demand testing where you can reduce costs by avoiding the necessity of making a big commitment to a test platform up-front. Many times we just do not know how many tests to budget for at the start of a new cycle.

b) Secure, Reliable Infrastructure: Since testing on the Cloud is mostly performed out of the enterprise's data center, the infrastructure has to be reliable and in accordance with the organization's security policies. Reliability is key since network,

server or firewalls can be a bottleneck to testing out of the data center.

Open-Source Testing Cloud Architecture:

The Cloud was mostly built on open-source technology. It is fitting that open-source testing is also built on open-source technology. Testing in the Cloud utilizes a distributed test environment.

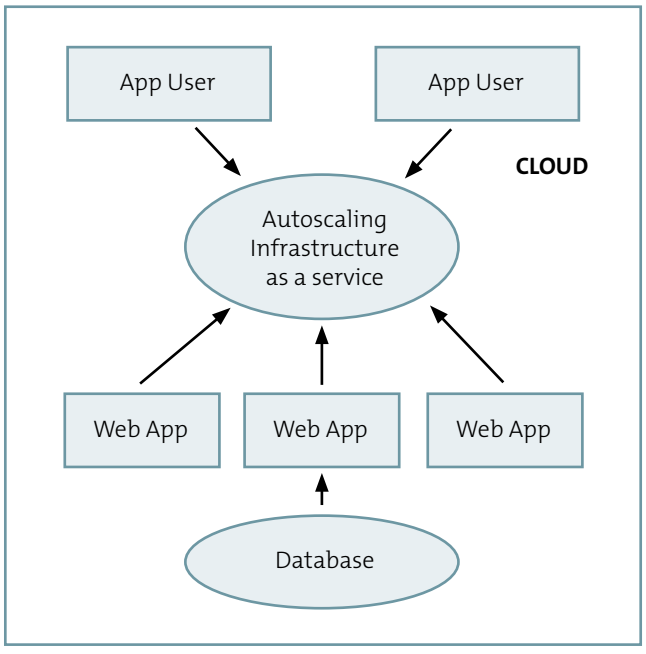
Many organizations are utilizing the distributed environment shown above to manage web application testing. A test operation console and runtime functions orchestrate tests authored in a variety of languages (Selenium, Sahi, SoapUI, etc.) as functional tests, load and performance tests and business service monitors in a distributed network of test nodes (also known as test injectors.) Users install and run the tests on test nodes in a grid of servers in a QA lab, in a Cloud test provider like Amazon EC2, or a combination of both. The approach is to create test nodes in an Iaas provider's instance which can run the test, retrieve the results, save the logged transaction times and record the success or failure of each step of the test to a log file. Finally the test console takes down the EC2 instances.

There are now businesses that provide Open-Source Test (OST) solutions. They provide support for open-source test authoring tools like Selenium, JMeter, Sahi, Windmill, Mozmill, and SoapUI in an integrated delivery platform.

Popular Iaas providers are Amazon EC2, Collabnet & Rackspace for testing using the Cloud.

Freedom Open-Source Software (FOSS) & Auto-scaling:

The Cloud moved forward in 2010, with many examples of process maturity now available . For example, Freedom OSS is an open-source environment that layers over Cloud-based environments to provide application auto-scaling. Auto-scaling allows you to automatically scale your infrastructure according to preset rules. With auto-scaling, the number of Amazon EC2 (or any other Iaas) machine instances scales up seamlessly during increased periods of load and scales down when the extra machine instances are not needed. Auto-scaling is particularly well suited for applications that experience hourly, daily, or weekly variability in usage.



A tool like Freedom OSS can successfully auto-scale the application by adding virtual application server nodes with increasing loads.

Auto-scaling is crucial for real business critical applications since real applications face varying and unpredictable loads. With auto-scaling you can, for example, create a rule to bring the application costs to zero, when your application is consuming low (or 0%) system resources.

Cloud Testing Benefits:

- **Run at user levels never before possible** - Building a test lab used to be prohibitively costly for medium and large scale load tests. Cloud computing delivers bandwidth and computing power to achieve tests in the millions-of-users levels at commodity prices.
- **Run from multiple locations around the world** - Cloud testing environments are a world-wide phenomenon. Open Source Testing (OST) solutions run tests in multiple Cloud testing environments. This makes it possible to manage performance from your users' locations.
- **Pay as you go, pay only for what you need** - Cloud testing eliminates the cost of building and maintaining a test lab for load and performance testing. No longer is there a need to provision expensive and difficult-to-manage quality test labs. Open-source Cloud testing enables you to pay as you operate tests.
- **Test in your environment, in a Cloud environment, or both** - Open Source Testing (OST) delivers ultimate flexibility.

Cloud Testing Risks:

The biggest risk with testing on the Cloud is that of operational test data security. Since the test data is being used outside of your data center, it can be a risk. Another risk could be that of the infrastructure itself. Unreliable infrastructure can result in network outages and other issues during test execution. Also, it is important to calibrate the test environment before testing on the Cloud. Search the Internet for „open source calibration testing" for popular and powerful methodologies and techniques.

Conclusion:

In this article we defined Cloud testing and we showed the benefits and risks of cloud testing. Open-source Cloud testing tools and processes reduce risks, increase security, and deliver service excellence. However, this is possible only if users ensure that the test framework and methodology in use support Infrastructure as a service (Iaas).

The value of open-source Cloud testing is not just in the compelling cost structure of the Cloud, but also derives from the auto-scaling and elasticity rules which permit organizations to control the load on an application. However, applications need to be tested after doing a careful risk-benefit analysis. Traditionally, if scalability has been a hindrance to your testing, you should be evaluating Cloud-based testing for future quality assurance needs.

Biography

Frank Cohen, Founder of PushToTest, Author of FastSOA. Through his speaking, writing and consulting, Frank Cohen is the expert that information technology professionals and enterprises go to when they need to understand and solve problems in complex interoperating information systems, especially Modern Internet Applications (RIA, Ajax, Flex/Flash,) Service Oriented Architecture (SOA,) and Web services. Frank is founder of PushToTest, the open-source test automation solutions business, and maintainer of the popular TestMaker open-source project. PushToTest customers include Ford, Sprint, Tribal DDB, Pepsi, and dozens of global enterprises. Frank is the author of several books, including FastSOA, the way to use native XML technology to build scalable service oriented architecture, and Java Testing and Design. For the past 25 years he has developed and marketed some of the software industry's most successful products, including Norton Utilities for the Macintosh, Stacker, and SoftWindows. Frank also organizes workshops on the web on open-source testing tools and methodologies. Register for a free workshop at workshop.pushtotest.com. You can contact Frank at fcohen@pushtotest.com

Saurabh Mallik is a Marketing Manager at PushToTest. Prior to joining PushToTest, Saurabh graduated with an MBA from the University of California, Irvine. He has worked as a test engineer with a large enterprise focusing on performance and load testing. His interests include Cloud computing, test automation & product marketing. He can be reached at saurabh@pushtotest.com.

Hardening Internet Business: Fighting Online Identity Theft and Fraud

by Shakeel Ali

In recent years, there has been a dramatic increase in the number of identity theft cases. At the core, this issue has not only threatened a number of online consumers to stop transacting over the internet, but has also affected retail sales of e-commerce businesses. While taking into account "a business security and satisfaction", it's not just the matter of stealing the financial data but the way it is being used after that by fraudsters. Implementing a secure network, integrating strong authentication methods, employing physical security controls, developing secure applications, providing regular security awareness training, and complying with PCI-DSS requirements, are some of the few known countermeasures taken by the online industry. However, psychologically, the human element of security can be reversed (e.g. social engineering) and the rules can be changed to manipulate and penetrate the existing security infrastructure. The points discussed below highlight the protection measures that should be practiced by the organizations relying on e-commerce platforms for their business. This will help them to reduce the threat of fraudulent transactions and detect any unauthorized and unusual attempts while making their business a safe and secure place for legitimate customers.

1. Log all the transactions and customer session information. The front-end e-commerce application should be designed with robust security objectives that could provide all necessary clues during a fraud investigation. These applications mainly include logging the customer activity, financial data, purchase location, shipping details, total amount charged, and the merchant used for transaction. The e-commerce platform should always be able to issues a unique key for each transaction in order to track the purchase in case of a claim.

2. The e-commerce application should be robust, secure, and capable of retrieving useful information during the customer interaction through the web browser, such as IP address, geographical location, proxy detection, time-zone data, and other HTTP headers (HTTP_X_Forwarded). This can seriously detect and deter the possibility of fraud. Based on logging and monitoring policy, we can easily track down the fraudster and prosecute any illegitimate act perpetrated against the business.

3. Detect the variation and speed by which the orders have been placed. There could be a malicious automated transaction being set off at particular session intervals, choosing the same product for each order, using different stolen credit card details, and operating behind multiple IP addresses. Such situations are unique and traceable by implementing the customer purchase trend tracking and analysis process. Using this process the customer service department would be able to identify an unauthorized illegal purchase. Moreover, the phone verification process can be an added value. For better security, we can implement Captcha and other consumer security authentication methods (e.g. Visa VBV, MasterCard Secure Code) which would fail these automated checks and in rare cases the human fraudsters as well.

4. Keep the fraud deterrent tactics secure and covert so that the fraudsters don't notice that you have detected them. Let all the transactions be processed as original and review them manually in order to detect suspicious items (e.g. shipping address, anomaly over completing the application forms). This policy can be segmented into various levels across the organization, such that the regular orders could be taken by service department personnel and then inspected by fraud specialist before being processed. This will ensure the integrity of a business and halt any fraudulent activity until being traced by the law enforcement agencies.

5. Never ignore the risk of fraud with mobile commerce. The rapidly increasing trend of using mobile devices to purchase via a website has not only put security in jeopardy, but has also presented exceptional challenges for security gurus. For instance, one can use VMware to emulate a Smartphone on the local computer and gain access to a mobile commerce website through a proxy. The detection mechanism is possibly harder to implement but, based on advance knowledge of the TCP/IP stack, one can make use of fingerprinting techniques to detect similar activities to those of a normal computer user. However, certain technology limitations would remain "challenging" and "irresolvable". Hence, manual inspection on all purchases is highly recommended.

6. Mask all the sensitive data in transit and save it in a secure location (e.g. choosing non-routable server IP address). The easiest way to handle it is by implementing secure cryptographic schemes and algorithms (e.g. SSL or TLS) on top of the e-commerce application. This should necessarily avoid the eavesdropping and hijacking of the financial data transferred in clear text.

7. Set up a monitoring policy for e-commerce applications to detect and reject random data input to the application form fields. This should prevent the attacker from disclosing and learning the fact of how the application behaves. In other words, your application should go through a secure development lifecycle in order to avoid common security vulnerabilities (e.g. buffer overflow, SQL injection, cross-site scripting, insufficient authorization, information leakage, denial of service condition, session hijacking, etc). In wider application environments you should deploy an intrusion detection and prevention system (IDS/IPS), or a web application firewall (WAF) to guard against application and network layer threats. Physical security would also remain the top concern over data retention, retrieval, and destruction policy.

The countermeasures discussed above do not highlight the technical details in-depth, but they can still be used to derive the business security strategy which should be followed by e-commerce application developers, security administrators, customer service departments, and other organization entities. Remember: Security is a fact and there is no escape. The coordination between multiple activities, such as network and system protection scheme, secure application development and deployment, regular security audits, security awareness training, and incident response policy should all contribute towards better and safer platforms.

References

Shreeraj Shah (2009), Application Defense Tactics & Strategies - WAF at the Gateway, HackInTheBox Conference 2009, Dubai.

Joan Goodchild (2008), [online] Symantec Threat Report: Three Ways Internet Crime Has Changed CSO Online Security and Risk, Available from:

http://www.csoonline.com/article/458170/symantec-threat-report-three-ways-internet-crime-has-changed [Accessed 18 September 2010].

Hamiel & Moyer (2009), Weaponizing the Web: More Attacks on User Generated Content, Black-Hat Conference 2009, USA.

Peter Guerra (2009), How Economics and Information Security Affects Cyber Crime and What It Means in the Context of a Global Recession, Black-Hat Conference 2009, USA.

Rajab, M., Zarfoss, J. (2006), A multifaceted approach to understanding the botnet phenomenon, 6th ACM SIGCOMM conference on Internet Measurement,

SESSION: Security and Privacy, 2006, pp. 41–52.

Biography

Shakeel is a CEO and co-founder of Cipher Storm Ltd, UK. His expertise in the security industry has provided marvelous benefits to various businesses and government institutions. He is also an active and independent researcher, who has been evangelizing security practices via articles and blog at Ethical-Hacker.net. Shakeel has assessed and measured the security of several business applications and network infrastructures for global organizations. The vision of his part in a collective interview conducted by President of OdinJobs (Careers section) gave clear highlights on skills, knowledge and experience required to deal with today's technical and managerial goals. Shakeel has recently partnered with BugCon conference 2010 to present the best of breed cyber security threats and their solutions, and industry verticals. This joint venture has attracted many audiences from various industry sectors, including government, education, media, commercial, banking and other respective institutions.

Masthead

EDITOR
Díaz & Hilterscheid
Unternehmensberatung GmbH
Kurfürstendamm 179
10707 Berlin, Germany

Phone: +49 (0)30 74 76 28-0 Fax: +49 (0)30 74 76 28-99 E-Mail: info@diazhilterscheid.de

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

EDITORIAL
José Díaz

LAYOUT & DESIGN
Katrin Schülke

WEBSITE
www.testingexperience.com

ARTICLES & AUTHORS
editorial@testingexperience.com

350.000 readers

ADVERTISEMENTS
sales@testingexperience.com

SUBSCRIBE
www.testingexperience.com/subscribe.php

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

ISSN 1866-5705

Index of Advertisers

EUROPE 2011

Testing & Finance

The Conference for Testing & Finance Professionals

May 9–10, 2011
in Bad Homburg (near Frankfurt am Main), Germany

The conference for Testing & Finance professionals includes speeches and field reports of professionals for professionals in the areas of software testing, new developments and processes. Futhermore there will be field reports for recent projects in financial institutions and theoretical speeches for regulatory reporting, risk- and profit based reporting.

Call for Papers

The Call for Papers ends by midnight on December 31, 2010 (GMT+1).

www.testingfinance.com/europe/en/

Exhibitors

iSQI International Software Quality Institute

Microsoft Visual Studio

Supported by

Díaz Hilterscheid

testing experience

VOB SERVICE

A Díaz & Hilterscheid Conference

Training with a View

Díaz Hilterscheid

"Simply a great course! Well worth it, when operating in the field of software testing.
A lot of knowledge is conveyed comprehendible in a short time."
Michael Klaßen, H&D IT Solutions GmbH

"Thank you for 3 very informative days.
I went home with new excitement for my job and passed the exam with 39 / 40 correct questions. "
Rita Kohl, PC-Ware Information Technologies AG

09.12.10–10.12.10	HP QuickTest Professional	Berlin
13.12.10–17.12.10	Certified Tester Advanced Level - TESTMANAGER	Berlin
10.01.11–13.01.11	Certified Tester Foundation Level	Berlin
17.01.11–19.01.11	Certified Tester Foundation Level - Kompaktkurs	Mödling/Österreich
19.01.11–21.01.11	Certified Professional for Requirements Engineering - Foundation Level | ENGLISH	Stockholm
20.01.11–21.01.11	Testmetriken im Testmanagement	Berlin
24.01.11–28.01.11	Certified Tester Advanced Level - TESTMANAGER	Berlin
07.02.11–09.02.11	Certified Tester Foundation Level - Kompaktkurs	Frankfurt
08.02.11–09.02.11	HP Quality Center	Berlin
09.02.11–11.02.11	Certified Professional for Requirements Engineering - Foundation Level	Mödling/Österreich
09.02.11–11.02.11	ISEB Intermediate Certificate in Software Testing	Berlin
14.02.11–18.02.11	Certified Tester Advanced Level - TESTMANAGER	Frankfurt
16.02.11–18.02.11	Certified Professional for Requirements Engineering - Foundation Level | ENGLISH	Helsinki
21.02.11–25.02.11	Certified Tester Advanced Level - TECHNICAL TEST ANALYST	Berlin
01.03.11–03.03.11	Certified Professional for Requirements Engineering - Foundation Level	Berlin
07.03.11–11.03.11	Certified Tester Advanced Level - TESTMANAGER	München
07.03.11–10.03.11	Certified Tester Foundation Level	Mödling/Österreich
09.03.11–10.03.11	HP QuickTest Professional	Berlin
14.03.11–17.03.11	Certified Tester Foundation Level	Berlin
16.03.11–18.03.11	Certified Professional for Requirements Engineering - Foundation Level | ENGLISH	Oslo
21.03.11–25.03.11	Certified Tester Advanced Level - TEST ANALYST	Berlin
28.03.11–01.04.11	Certified Tester Advanced Level - TECHNICAL TEST ANALYST	Mödling/Österreich
29.03.11–31.03.11	Certified Tester Foundation Level - Kompaktkurs	München
30.03.11–30.03.11	Anforderungsmanagement	Berlin
04.04.11–08.04.11	Certified Tester Advanced Level - TESTMANAGER	Frankfurt
05.04.11–07.04.11	ISEB Intermediate Certificate in Software Testing	Berlin
11.04.11–14.04.11	Certified Tester Foundation Level	München
11.04.11–12.04.11	Testmetriken im Testmanagement	Berlin
11.04.11–15.04.11	Certified Tester Advanced Level - TECHNICAL TEST ANALYST	Stuttgart
18.04.11–19.04.11	Testen für Entwickler	Berlin

more dates and onsite training worldwide in German, English, Spanish, French at

http://training.diazhilterscheid.com/

training@diazhilterscheid.com

iseb Accredited Training Organisation

IREB International Requirements Engineering Board

ISTQB Accredited Training Provider

- subject to modifications -