



THE TESTING PLANET

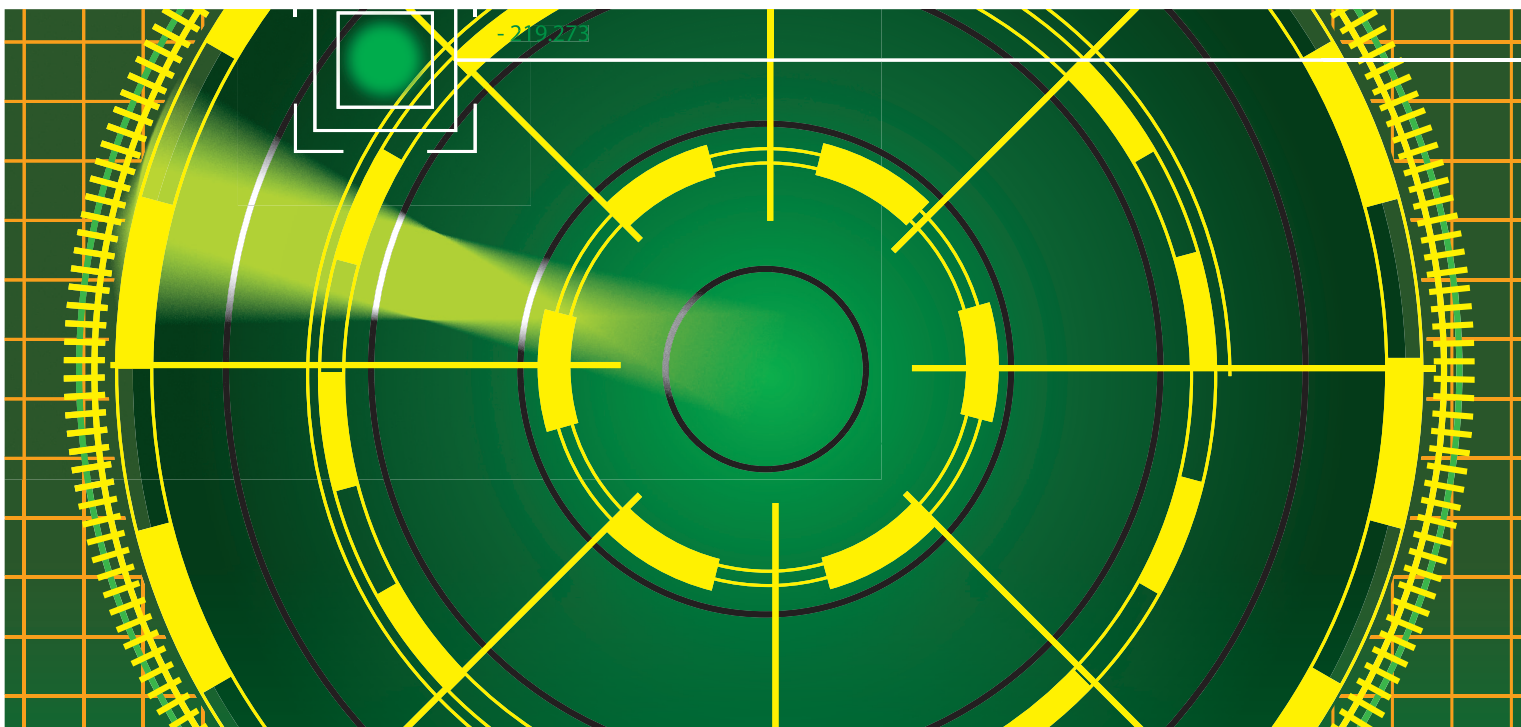


March 2012 | www.thetestingplanet.com | No: 7

£5

We take a look inside Sauce Labs

The San Francisco-based company talks automation, security and high-performance infrastructure - see page 12



Could TestOps mean a dramatic change to the testing profession?

Going mobile: Testing beyond the device

By Jonathan Kohl

Have you noticed a shift in technology lately? If you haven't, take a look around you next time you go to a restaurant. You will probably see people using their smart phones or tablets: fingers tapping away, eyes staring at the screen, an eerie blue lighting up their face. They are focused, engrossed, addicted, and not to the food, company and atmosphere in the restaurant. They are interacting with (and addicted to) applications on mobile devices that are convenient to carry with us wherever we go.

Mobile devices are our contact points to applications, services, the internet, and increasingly, our social lives. For example, consider that trip to a local restaurant and how mobile technology has become enmeshed in that experience. How many of the following activities can you relate to?

- Searching for a restaurant nearby (using location-based services or GPS.)
- Selecting a restaurant based on cuisine, location or price.
- Reading user reviews and ratings to help make a decision.
- Plotting out and following directions to the restaurant on a map.
- After arriving, checking in to a social networking application, alerting people in your network of your current location.

Continued on page 4

The future of software testing Part two - TestOps

By Seth Eliot

In the November 2011 edition of The Testing Planet, I made several predictions of the future of software testing,

- In part 1, I made the case that Testing in Production is the natural way to test software services, using real users and real environments to find bugs that matter.

- In part 3, I will explain how the Cloud will continue to gain traction and enable testers to exercise unprecedented effectiveness and productivity.

The subject of this article is what I call TestOps, and how it represents the potential for dramatic change to the testing profession and how software organisations test software and organ-

ise to assure software quality.

Software Testing Presently

To see the future we must first understand how most organisations test software in the present. Figure 1 is a simple but useful representation of this current state of affairs.

Continued on page 2

ALSO IN THE NEWS

SET THE LOAD TEST TOOL TO RAMMING SPEED

The vendor's salesman/trainer sat next to me at the conference table...
Continued on page 7

THE BUILD-A-TESTER WORKSHOP

The Testing Planet and the Software Testing Club previously introduced...
Continued on page 14

CROSSING THE TESTING MINEFIELD

You are with a party of friends and on a beautiful walk in the countryside...
Continued on page 21

THE EVIL TESTER QUESTION TIME

More provocative advice for testers who don't know what to do!
Continued on page 24



LETTER FROM THE CO-EDITOR



Wow! Where has the time gone? It seems like only a very short time ago since I accepted the role of The Testing Planet Editor, and here we are with my first issue. And in a new Premium format no less! It's great to be part of a team of passionate and dedicated Testers committed to producing the highest quality publication to inform, educate and of course entertain the testing community.

It's fair to say – when I took the job on, I didn't realise just how much work was going to be involved in bringing it all together. Or how many late nights would be spent reviewing content and responding to emails. Or how much of a push would be required to bring it all together during the final design and pre-publication stages. It's been a <<fascinating/exhilarating/frustrating/[insert adjective of the day here]>> journey and, if you're reading this – WE MADE IT!!!

Of course, time pressure and that all important final push to get your product shipped are part of the testing creed, right? I don't imagine any of us are strangers to having our test schedule squeezed to virtual non-existence, late nights and weekends working to finish that all important, final round of tests before we give the all clear.

But what about the time pressure to stay on top of your game, and keep abreast of changes in test approach, tools and technologies? How do you cope with that?

A quote that is still forefront of my thinking from 2011 is that testing well is all about learning rapidly (James Bach, Rapid Software Testing [course]). Testing is changing, maturing, and evolving all the time, and we need to evolve with it. Reading The Testing Planet is just one way that you can stay in touch with the thinking and development of our community of Testing Professionals. We invest our time and effort into bringing you the best thinking and writing, from the best testing minds so that you can benefit from their experience, and hopefully become a better tester as a result. We hope you enjoy it!

Simon Knight

Main story continued from page 1

Tests are run against the product and the product reacts. If this result matches the expected result (the oracle) it passes... it's green. A tester can look at all the passes (greens) and the fails (not-greens) then make pronouncements about the product quality. Distilling this down to three stages we get Table 1.

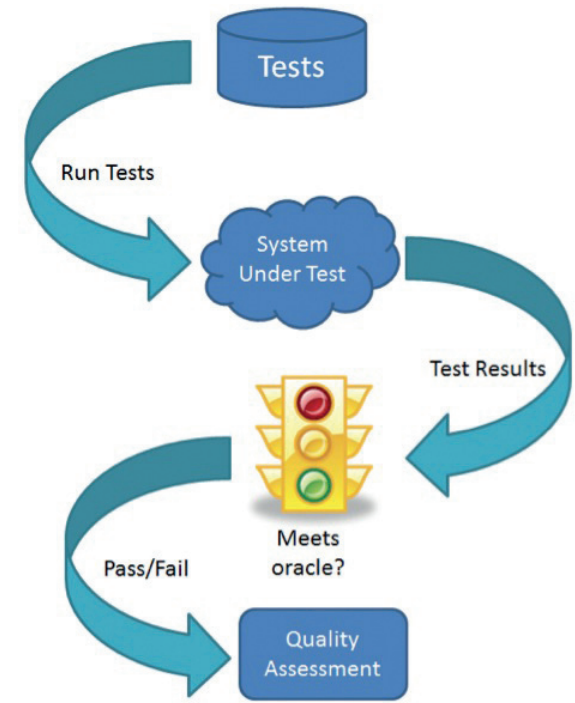


Figure 1. The traditional model of software testing

Tests are run against the product and the product reacts. If this result matches the expected result (the oracle) it passes... it's green. A tester can look at all the passes (greens) and the fails (not-greens) then make pronouncements about the product quality. Distilling this down to three stages we get Table 1.

Stage	Traditional Testing
Input	Run Tests
Signal (Output)	Test Results
Observe	Pass/Fail

Table 1. Three stages of traditional Test

Test results are our output, our signal to be processed and interpreted. However in the future we need to look more at a new signal. Software

services running in data centers have relied on Operations or “Ops” teams to run the data center. Ops provisions hardware, configures networking, then monitors and repairs any production impacting issues. Now, like the Ops team, testers will need to have more focus on the live site. Our new quality signal will come from this live production data, hence the name TestOps.

Products of the Future

To understand the future of software testing, look where the growth in software products is taking place:

- Google executes more than 3 billion searches, 2 billion video replays and absorbs 24 hours of video per minute ¹
- Microsoft Bing has grown from 10% of U.S. Searches in September 2010 to over 30% as of April 2011 ²
- Facebook has about 800 million active users sharing 30 billion pieces of content per month ³
- Amazon has more than 120 million user accounts, 2 million sellers and 262 billion objects stored in its S3 cloud storage system ⁴
- Twitter reached its one-billionth tweet after just 3 years, 2 months and 1 day ⁵

These successful, growing products are all services. In the The Future of Software Testing, Part 1 (November Issue – The Testing Planet) ⁶, I discussed how services allow us to Test in Production by giving us immediate access to production, where we examine and change what is happening there. But to see what else services give us that will change the way we test, consider these statistics:

- At Google system performance is tracked by their “Dapper” tool. More than 1 TB of sampled trace data per day and all data is kept for two weeks ⁷
- Facebook’s logging framework Scribe collects approximately 25TB of data daily ⁸
- eBay collects approximately 50 TB of incremental data every day to perform analytics ⁹
- Twitter stores 12 TBs of new data daily ¹⁰
- Microsoft online properties Bing, MSN, and Ad-Center collect and process 2 PB of data per day ¹¹

What services give us are telemetry data from users and the systems they are using, and in the cases above it is big data. Changing our signal to assess

Continued on page 3

AUTHOR PROFILE - SETH ELIOT

Seth Eliot is Senior Knowledge Engineer for Microsoft Test Excellence focusing on driving best practices for services and cloud development and testing across the company. He previously was Senior Test Manager, most recently for the team solving exabyte storage and data processing challenges for Bing, and before that enabling developers to innovate by testing new ideas quickly with users “in production” with the Microsoft Experimentation Platform (<http://exp-platform.com>). Testing in Production (TiP), software processes, cloud computing, and other topics are ruminated upon at Seth’s blog at http://bit.ly/seth_qa and on Twitter (@setheliot). Prior to Microsoft, Seth applied his experience at delivering high quality software services at Amazon.com where he led the Digital QA team to release Amazon MP3 download, Amazon Instant Video Streaming, and Kindle Services.



Continued from page 2

software quality from test results to telemetry data is a game changer for testing. Even if your service does not generate big data, any product operating at internet scale is going to see usage and data of sufficient magnitude to assess quality.

A New Signal for Quality

Let us update our three stages of test. We’ve determined that our signal is going to be the telemetry or big data coming from the system under test. Then the input can be the real world usage of the product. Then what about the observe stage? Data by itself has little value until it used or transformed to find patterns or prove hypotheses. Such analysis can be done to calculate Key Performance Indicators (KPIs) which are then compared to target requirements. Or we can look for patterns in the data that give insight into the behaviors of users or systems. These stages are summarized in Table 2.

Stage	Traditional Testing	TestOps
Input	Run Tests	Production usage
Signal (Output)	Test Results	Telemetry Data
Observe	Pass/Fail	KPIs and Patterns

Table 2. Three Stages of Traditional Testing and TestOps

We can start by using real production usage to decide where to look. For example with Bing, we can identify the top 100 query types used and conclude that we might get the most value by starting to look there. Among the top queries are those for stock prices. On Bing the query “Amazon Stock Price” should return the Finance Instant as top result. Figure 2 shows what actually happened with one release, as well as what the missing Finance Instant Answer should look like.

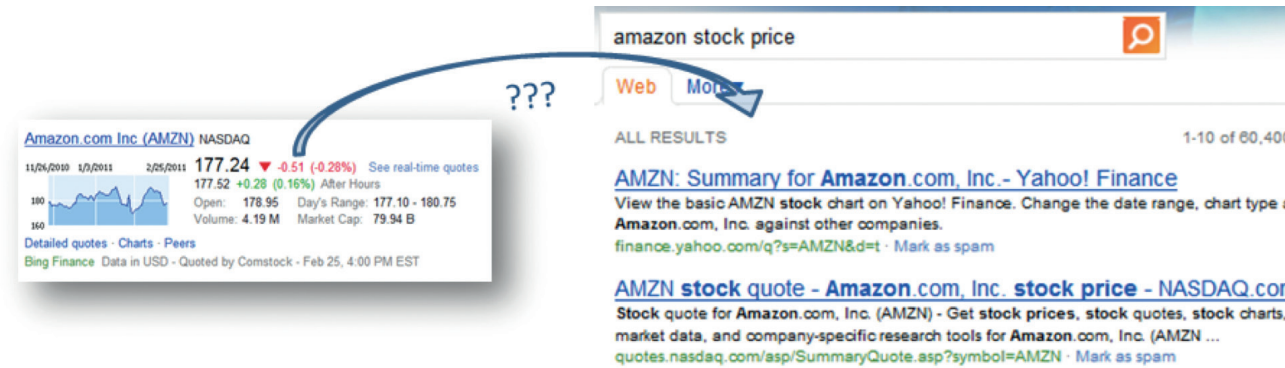


Figure 2. A Bing bug....Amazon stock query missing Finance Instant Answer

A classic test approach might be to iterate through the S&P 500 or the Wilshire 5000 and execute the query “[company_name] stock price”, substituting in each company name and searching for where the instant answer did not fire. We would then find that the following test cases pass:

- Expedia stock price
- Comcast stock price
- Bed Bath and Beyond stock price

All-pass is not all good in this case as there are bugs we still have not found. Therefore Bing implemented Bug Miner¹² which enables testers to configure queries or patterns and see what results real users are getting. Using real user data Bing found and fixed these bugs (did not return the Finance Instant answer) that traditional testing would have missed:

- expedia.com stock price
- Comcast Corporation stock price
- Bed Bath & Beyond stock price

A consideration when using telemetry data as your signal is that you may have different signals from different users on different products. The example in Figure 2 would still seem to “fail” in the UK, but the international sites do not use the same logic as the US site. Different regions and different products will have differing signals that each must be interpreted in the context of those respective products.

Another example of using telemetry data as the signal for quality is Microsoft Hotmail. By instrumenting their web pages they get anonymised signals telling them when a user performs actions like opening or sending a mail for example. Using this data, testers can calculate how long it takes across millions of users to perform key tasks, and interrogate this data to see how Hotmail performs across different operating systems and web browsers. By using this information key bottlenecks can be identified. In one example Hotmail re-architected image size and static content to improve upstream traffic, improving performance by 50%.

More Ways to Generate the Signal

Monitoring and analyzing production usage to access the telemetry data signal for software quality is a very approachable way to implement Testing in Production. But there are also more active steps we can take to vary the input stage of our three step approach and gain quality insights.

KPIs. For a scenario did we meet the “five nines” (99.999%) availability? Or did we complete the task in less than 2 seconds 99.9% of the time? The signal is still the same, it is still the data coming out of our system under test, but we are triggering that signal with our testing. Exchange Online runs thousands of tests constantly in production alerting them to small non-customer impacting outages which represent regressions and risks to the service¹³. They find quality issues before the customer does.

Another way to vary our Inputs is to deploy the new version, feature, or product change so only a sub-set of users sees it. This is called Exposure Control. Using Exposure Control we can run A/B tests, comparing the data signal from the old version (A) to the new version (B). Google, Amazon, Microsoft and others regularly do this. Often UI changes are assessed based on business metrics, but testers can also make use of back-end changes “under the covers” to assess quality metrics. Google regularly tests changes by limiting exposure to explicit people, just Googlers, or a percent of all users¹⁴.

So we can now update our testing model diagram to represent this new quality signal, and the three inputs to drive it in Figure 3.

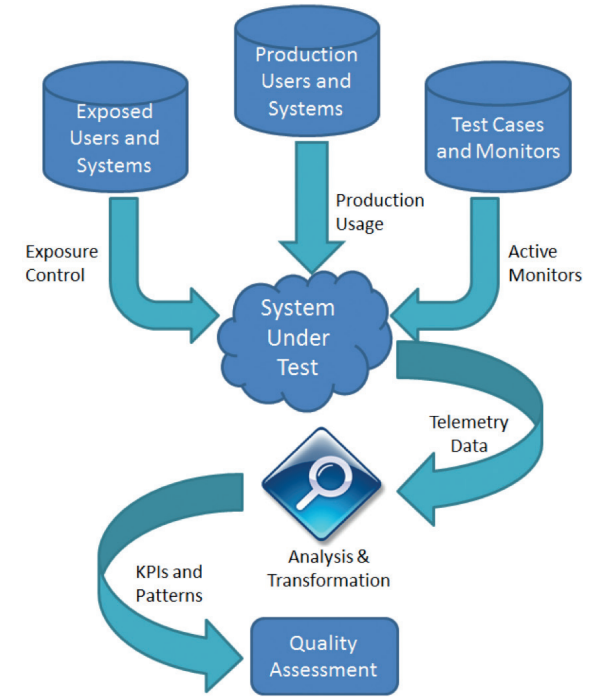


Figure 3. The TestOps model of software testing

TestOps? What Does This Mean for Me?

Just the change in signal alone indicates a big shift in the testing profession and how we test software. But what does TestOps mean to the day-to-day work of the tester?

The roles of the Developer and Tester begin to change, with each looking more like the other:

- The tester’s focus can be much less on the up-front testing and running of discrete functional test cases. Developers have always had responsibility for unit testing, but to enable testers to focus more on production and high context scenarios consider moving up-front functional testing to the developers. Put another way, a developer must use every available

Continued on page 4

Continued from page 3

- means to produce code free of low context bugs that can be found by running on his desktop.
- Testers now need to focus on a Test Oriented Architecture¹⁵ that enables us to collect and identify the right data signal. Testers need to create tools to run the tests in production, monitor the data signal, and analyze this signal. These are roles that look very dev-like.

The roles of Operations and Tester begin to change, similarly with each looking more like the other:

- Testers are now focused on the live site- the traditional domain of operations. The tester’s skill is now applied in determining the connection between the data signal and product quality.
- Whether scanning the production data signal or firing synthetic transactions, the tests we write now look a lot like monitors. Something Ops has been doing all along. The value test brings is in moving these monitors from heartbeats and simple scenarios to high context user affecting scenarios.

Across the major services these changes are already taking place. At Facebook and Microsoft Bing they practice combined engineering, with no separate test discipline. The goal is for developers to not only develop code but to also do all of the TestOps tasks above. This can be a difficult model to execute and will fail if engineers revert to “classic” developer responsibilities alone thus abdicating quality. A separate TestOps role can ensure quality is

foremost even while blending this test role with Dev and Ops. Therefore Google, Amazon, and Microsoft services (other than Bing) instead maintain their test disciplines, but with high Dev to Test ratios from 3:1 to 8:1. A 1:1 ratio is not necessary as testers are no longer writing rafts of tests for each discrete feature and are instead focusing on tools and processes to harness the power of the data signal. Companies like Amazon.com and Google maintain central teams to further put Developers’ and Testers’ focus on the live site creating tools to make deployment and monitoring easier.

It’s All Test, It’s All Good

Yes, Testers can still test up-front. Black box and exploratory testing still add value and the traditional test model can still be applied. But depending on your product - especially if it is a service - the TestOps model using the big data signal can be your most powerful tool to assess software quality. In Test is Dead¹⁶ Google’s Director of Engineering made the case that the data signal tells us if we built the right thing and that traditional testing is much less important. Test is not dead. The power of big data and the data signal will also tell us what we need to know about quality - not just did we build the right thing, but did we build it right. □

REFERENCES

1. <http://searchengineland.com/by-the-numbers-twitter-vs-facebook-vs-google-buzz-36709>
2. <http://mashable.com/2011/04/11/bing-search-growth/>

REFERENCES CONTINUED...

3. <http://www.facebook.com/press/info.php?statistics> http://www.businessweek.com/magazine/content/10_49/b4206039292096.htm
4. <http://blog.twitter.com/2011/03/numbers.html>
5. <http://www.thetestingplanet.com/2011/11/november-2011-issue-6/>
6. <http://research.google.com/pubs/archive/36356.pdf>
7. <http://www.infoq.com/presentations/Scale-at-Facebook>
8. <http://www.slideshare.net/RandyShoup/more-best-practices-for-largescale-websites-lessons-from-ebay>
9. <http://www.slideshare.net/raffikrikorian/twitter-by-the-numbers>
10. http://research.microsoft.com/en-us/events/fs2011/helland_cosmos_big_data_and_big_challenges.pdf
11. Edward Unpingco; Bug Miner; Internal Microsoft Presentation, Bing Quality Day Feb 11, 2011
12. Experiences of Test Automation; Dorothy Graham; Jan 2012; ISBN 0321754069; Chapter: “Moving to the Cloud: The Evolution of TiP, Continuous Regression Testing in Production”; Ken Johnston, Felix Deschamps
13. Google: Seattle Conference on Scalability: Lessons In Building Scalable Systems, Reza Behforooz <http://video.google.com/videoplay?docid=6202268628085731280> [timestamp: 20:35]
14. <http://www.setheliot.com/blog/bsc-east-2011/>
15. <http://youtu.be/X1jWe5rOu3g>

Second story continued from page 1

- Searching the web to answer questions about the items on the menu. (I always forget what aioli is.)
- Translating a meal that is in a foreign language.
- Checking the nutritional information for a meal.
- Once the food arrives, taking a picture of it and uploading it to your social networking profile.
- Friends commenting about the photo and your meal.
- Throughout the meal, posting and responding to comments about the meal on your social networks.
- At the conclusion of a meal, posting positive, neutral or negative comments about the experience to restaurant review applications.
- If the experience was poor, ranting on public social media.

We haven’t even touched on the buzzing and beeping that occurs while receiving emails, text messages, and phone calls. Just imagine the technology that is required to support all of these activities. If you dissect it, the infrastructure and underlying complexity is amazing. The mobile device is just our primary contact point to a large amount of technology that supports our mobile-enhanced, social and work activities. All those photos we upload and share, all that data we have access to and send to our social networks, all have to be stored somewhere and quickly accessible not only to us, but to those in our

networks. This provides new challenges from a testing perspective.

The people who are using these applications are using them on the move. They interact with them physically, in different environments with different underlying infrastructures and in all weather. As testers, we need to interact with the application, find things that are difficult to input into, and to see, and find problems that are caused by being in interesting locations. These are the problems your end users will encounter the most, and will frustrate them the most. (I have developed a framework to help with testing these kinds of activities called I SLICED UP FUN!¹ However, merely testing the front end of many mobile applications ignores the support infrastructure underneath that allows us to work with the devices the way we do. We also need to analyze that and figure out potential weak points that we can exploit when testing. Two areas that have testing implications in the mobile space are data and the cloud.

Most testers are familiar with data storage, and software interactions with data-related software and hardware. If you aren’t familiar with “the cloud”, they are machines that are available for use for different purposes for a fee. You essentially rent time, space and processing power from a company that provides you with access to their machines over the internet. It’s a relatively inexpensive way to add processing power, distribute access and store and distribute data. There are also GPU (graphics processing unit) cloud centers that can be used for graphics rendering, and movement in areas like grid comput-

ing to help distribute processing across a wide array of machines.

Data storage and hardware/software/network interactions can have an impact on performance. Using the cloud is a powerful tool, but I’ve had the following challenges when testing software on the cloud:

- You lose the complete control you have over your own equipment.
- Cloud services may not have the uptime you are used to.
- Unforeseen network interruptions may occur.
- There are service outages, and bugs that cause longer, unplanned outages.

These issues can cause time and frustration when trying to track down performance issues or timing-related bugs.

Here’s an example: a team I was working with had developed a mobile application on several popular smart phone and tablet platforms. They already had a web infrastructure in place for a web-based application, but they exploited mobile devices to help put media in people’s hands. They were able to use features from mobile devices to enhance this experience, and provide connectivity and productivity for remote workers who were often away from desktops or laptops. The back end infrastructure was a typical web application using the LAMP stack (Linux OS, Apache web server, MySQL database

Continued on page 5

Continued from page 4

and PHP as the programming language). This worked well for web applications, but they weren't sure how the extra load of more and more people connecting and downloading and uploading media files would stress the system. I joined the project team to help with performance and functional testing.

We tried using load generation tools to simulate the extra web traffic, but this didn't address the media downloading and interactions on mobile devices. To simulate this effect, we used functional test automation tools to load the application in a web browser, sized to the same size as the mobile screen, and simulated user input by sending messages directly at the protocol level. We then used virtual servers and an internal cloud infrastructure to generate load across multiple machines. After jumping through a few technical hurdles, we were able to remotely execute tests over several machines on different networks, with about eight simulated mobile clients on each machine. This effort revealed some astounding results. The server behaved quite differently with the mobile client than with the old web client, and some of the simulated mobile clients suffered from serious performance and functional problems.

Once these issues had been sorted out, the process was reversed. Now that we knew some of the common trouble spots, and were able to simulate those conditions on the server, while testing with a small number of mobile devices to see how the real thing handled server performance problems.

After we felt confident that we had dealt with most of the performance problems using both approaches, we pushed out builds to our beta testers and had them re-test the application in the field. Some of them still reported serious performance issues, and a pattern emerged: the people who were the furthest away from our data center had the worst performance.

Data Implication: Distributed Data with the Cloud

We discovered that when using large media files, people who were furthest away from the data center experienced the poorest performance. It turned out that the combination of lower processing power on the

AUTHOR PROFILE - JONATHAN KOHL

Jonathan Kohl is an internationally recognized consultant and technical leader. Based in Calgary, Alberta, Canada he is the founder and principal software consultant of Kohl Concepts, Inc. Jonathan helps companies define and implement their ideas into products, coaches practitioners as they develop software on teams, and works with leaders helping them define and implement their strategic vision. He is also a popular author and speaker. As a thought leader in mobile application testing, exploratory testing, developing policy and strategy, and helping teams adjust to methodology changes, Jonathan doesn't just write and talk about developing software, he actively helps teams deliver the best products they can.

mobile devices compared to desktop computers, coupled with the complexity and differing speeds of data networks made performance problems much more apparent. Performance issues you barely noticed on a desktop were unbearable on a mobile device.

We needed a quick fix, so we looked at what we had done to design tests: we decided to use the cloud. We were already using cloud services for some of our media storage, so we expanded those services to data centers that were closer to the geographic areas where users experienced poor performance. This data distribution using different cloud centers worked well. This was a bit of a shock for us, but if you model the system from storage to download and interaction, it begins to make sense.

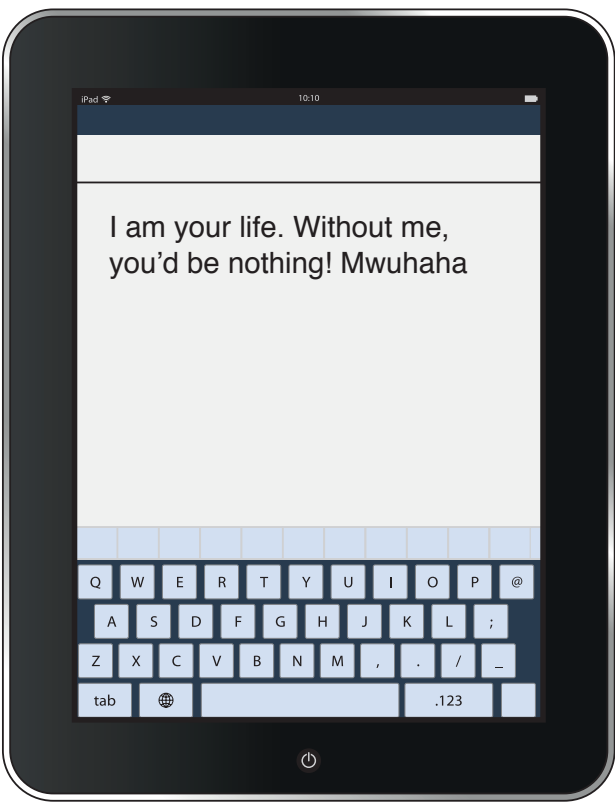
Data Implication: Reduce Bottlenecks with noSQL

Social networking infrastructures, and systems with high volume over a large geographic area, have highlighted interesting data interaction bottlenecks. If you look at a typical web application such as a LAMP stack, you tend to have a synchronous flow to and from the database, and most of us use some sort of Entity-Relationship Model and SQL to access the data. (Even if the SQL is generated by an Object-Relational Mapper.) When you have many reads and writes with a large amount of data, with different circles of people who require instant, simultaneous access, you inevitably run into bottleneck issues, which translate into poor performance at the end user level. Many social applications store an enormous amount of data, with users posting pictures, video, sound files and other items in their systems. Add more users with the ease of use that mobile devices provide, and bottleneck and performance problems become worse.

noSQL² is a movement that gained popularity, in part to help deal with some of these problems. Social networking systems have different data usage and requirements than other applications, so as programmers tried to solve these problems by adding parallel database interactions, different hardware and software solutions, they inevitably began to challenge the Entity-Relationship Data (ERD) model itself. What many found is that the ERD model we know and love does not always work for social data with its various relationships, and constant interaction. Without boring you with technical details, noSQL³ is a different model to save, access, update and read data than a typical database. There are different technologies used, many of them are also built to scale rapidly and cope with the rapid growth and massive data requirements that social and similar systems depend on.

From a testing perspective, this means we have to change the way we interact with data, and challenge some of our assumptions about how it is stored. noSQL systems may use something that looks like a database, or they may store data in different sorts of files. Querying may use something altogether different than SQL. One of my favorite SQL alternatives is XQuery's FLWOR⁴ (flower) expressions. We also have to look at configuration and performance, with less testing tool support than what we enjoy with traditional, ERD systems.

Many noSQL tools are newly developed, but the storage principles behind them are the same as any other persistence model. You will need to adjust how you think about storage, access and learn different tools.



Bringing it All Together

In many cases, the application we interact with on our mobile device represents the thin end of the wedge, or the tip of the technology iceberg. If we run our applications in ideal environments under ideal conditions, and ignore the infrastructure they depend on, we will miss important bugs. The problems our end users run into aren't necessarily that easy to discover if we are sitting in our test lab under ideal conditions.

What happens to the winter user who tries to use your device with gloves on? What problems might someone in a different country encounter? What does our back end infrastructure look like? Does it depend on new technology? How might it perform and handle data uploads and downloads seamlessly? What complications are introduced by the size and type of data our users require our system to handle? How might their physical location in relation to our data center have an impact?

As mobile testers, we need to look at the entire system and come up with creative ways to simulate real-world conditions using different kinds of tools and techniques to help discover the problems that are important to our users. In this article, we've looked at two specific areas: data and the cloud, but there are more. When testing mobile devices, research and look beyond the device to get the full picture. Only testing from the application-level may not be enough. Testing mobile applications with the full picture in mind will lead you to more of the important problems that your users are likely to encounter. □

REFERENCES

1. <http://www.kohl.ca/articles/ISLICEDUPFUN.pdf>
2. <http://radar.oreilly.com/2012/02/nosql-non-relational-database.html>
3. http://www.strozzi.it/cgi-bin/CSA/tw7/1/en_US/nosql/Home%20Page
4. http://www.w3schools.com/xquery/xquery_flwor.asp

A cautionary tale regarding the importance of exercising care when choosing mobile device test data

(a.k.a. be careful who you share your address book with)

By Stephen Janaway

Having accurate test data is important. Of course you know that already, it's one of the first things you are taught as a tester. Data, data, data; there are many articles in this publication and in blogs across the internet that can tell you that. "Make sure that your data reflects your real users", they say. "Make sure that you keep that data up to date". "Make sure that you have as much data as possible".

For those of us testing mobile devices and applications then, whilst all the advice above is true, there's more to take into account.

You Need Real Numbers

The first thing you quickly learn when you start to test mobile devices is that you need to test on mobile devices. Whilst that may sound obvious, there's a little more to that statement than meets the eye.

Real devices mean one thing – real subscriptions. And real subscriptions mean real SIM cards and real SIMs normally mean real cost. Of course you can consider outsourcing, or you can consider using services such as DeviceAnywhere¹, and these are certainly worthy of consideration, but at some point you really need to spend some money and buy some phones or tablets. And you need real phone numbers to test with.

You need real numbers and real address books full of other people's real numbers. This is especially important when testing either applications that use contacts and phonebooks, or applications which test the ability of the device to make or receive calls. The interruption case when your application stops or suspends when a call comes in is a very common place to find bugs. The way the phone reacts when you stop your application to place a call is equally bug ridden. To make calls you need real numbers.

Make Sure Your Numbers Are Your Own

OK, so you've done your homework and you've got some SIM cards and you're ready to test. But first you need an address book in the phone and we all know that when it comes to filling in a database

AUTHOR PROFILE - STEPHEN JANAWAY

Stephen Janaway has been working in software testing for over 12 years, focusing on the mobile device and applications area. He's worked for companies such as Ericsson, Motorola and Nokia, in software testing and test management roles, and is currently heading a software test and quality assurance group in Nokia UK, focusing on mass market devices.

Stephen blogs on his favourite subjects (software testing and Agile techniques, particularly Kanban) at www.onegreenhill.blogspot.com and you can follow him on twitter @onegreenhill.



with test data then the more the merrier. So why not just fill it with that handy backup you made yesterday from your own phone? After all, you're only going to test this yourself and you only need the data this afternoon.

Or maybe the Test Manager has just popped by your desk. "Just give this a quick test of your phone" they'll say. "You are the only one in the office with a Windows Phone". "Oh and I need this done today".

Stop. Consider this real world scenario that comes from experience. You needed that phonebook filled quickly and the personal backup was just too tempting. So you go ahead, you fill it up and you start testing. Hang on, there's a bug, that doesn't look right when your application displays user data from a full phonebook.

Right, it's time to raise a bug. Add the details and attach the test data, time for a quick coffee and then it's time to go home. Job done?

Well yes and no. You have raised the bug. You have explained the failure case in detail for the developer to analyse and you've given them the data to use. Good Job. But wait a minute.

You've just handed out your entire address book to the rest of the company. It will go far. Not deliberately, but good real-world test data is difficult to find. So people will use this newly found gem. They'll download it from the defect management system and they'll store it offline. Before you know it, one of the automation team will have started to use it. They probably found it when they started adding the failure case to the regression test suite. They don't know it's your address book; it's just test data to them. Suddenly your Mum starts receiving 20 calls a day from an unknown number, and there's no-one on the other end when she answers. She's not happy...

Does This Really Happen?

Yes. The story above is a real one. I've seen personal phonebooks passed around in bug reports, I've seen them turn up in automated test scripts and I've seen them used (with the best intentions) in a whole lot of places that they were never intended to be. I've seen them disappear, only to return a few years later in test reports from outsourcing companies. This is a cautionary tale. Treat what's yours very carefully!

Auto Generation Will Solve This, Right?

Well yes and no. Auto generation of data such as phone numbers is great when you just need realistic data in the right format and size. Be careful, especially when auto generating the phone numbers. It is very easy to find that you have auto generated real numbers. And then you have the same problem as before, except you don't know who is accidentally being called.

So What Are the Other Options?

Whilst auto generation is ok, there are times when only a real number will do. For this you need to either invest in a selection of SIM cards and associated subscriptions, or you need access to a pool of devices.

Often a combined approach works. Auto-generate as much data as possible (being careful) and supplement this with a small selection of real data, which you control. Larger companies, particularly in the mobile network vendor space also have entire closed phone networks, which means they can allocate as many SIMs and numbers as necessary. But for everyone else, Pay As You Go, pre-paid subscriptions are particularly cost effective and can ensure that when you do not need them that they are not costing you anything. You should work out the right number for you, taking into account that it's always going to be one more than you think.

The Conclusion of This Cautionary Tale

Be careful with what you have. Make sure you keep your personal data safe, and don't take the easy route when you are testing that application, or when the boss says "just give it a quick go with your phone, will you?" Make sure your test data is exactly that. Otherwise pretty soon you can find that you've shared far more than you were meaning to and once it's gone then it's pretty much impossible to get back. □

1. <http://www.keynotedeviceanywhere.com/index.aspx>



Set the load test tool to ramming speed!



AUTHOR PROFILE - BRIAN J. NOGGLE

Brian J. Noggle has worked in quality assurance and technical writing for over a decade, working with a variety of software types and industries. He currently works as a freelance software testing consultant through his own company, Jeracor Group LLC and has recently published a novel set in the IT world, John Donnelly's Gold.

By Brian J. Noggle

The vendor’s salesman/trainer sat next to me at the conference table, his laptop open between us with the software package that my employer and his chief developer had selected before they hired me as their professional quality assurance person. The vendor was demonstrating the load testing capabilities of the package, how it could take a regression test script and run it with simultaneous virtual users to create a load test. He showed me how to establish the ramp-up time, wherein I could add five users every thirty seconds to gently increase the load on the application under test.

He slid his laptop closer to me and let me give it a try. I set the virtual user number to the licensed maximum, 200, and set the ramp-up time to 0, and then I clicked the button to launch the test, bringing a virtual thunder strike of load against the Web application.

“What are you doing?” the seller whimpered, sounding not unlike the examiner during my first driver’s license test.

What was I doing? I was certainly ignoring the white-gloved niceness of performance testing, where one uses the tool to generate a gentle upward slope or step line of user load. Performance testing that allows one to nicely project how the delicate little application will perform with 25 users on a sunny Saturday afternoon in late May, when the winds are low and the cumulous are fluffy harbingers of continued good weather.

Your performance testing strategy needs not only to account for a nice, gradual increase in Web traffic and to create a sense of well-being in knowing that, in optimal circumstances, your Web application won’t falter. Your testing strategy must also challenge your application as much as possible, up to and including launching all of your virtual users at once, even targeting all virtual users at a single function instead of using “realistic” user scenarios.

Once your application is live, any number

of circumstances will mimic that load testing assault except that the user load will probably exceed your virtual user licenses. Many scenarios might arise, often without input from the IT staff developing or deploying the application that can suddenly and dramatically increase traffic to your application. In many of these circumstances, your organization’s IT staff might only know about the traffic spike when it happens.

Circumstances Outside the IT Realm Change

Business considerations outside the IT world can lead to a sudden spike in user activity. Recently a backend processing provider for health insurance prescription benefits ended its relationship with a large pharmacy chain in the United States.¹ As a result, other pharmacies began transferring many millions of prescriptions records, many of them in the first few weeks of the year. The hundred million or so records represented an addition to the normal daily transactions.

My pharmacist reported that was down for a whole day during the first weeks of January when angry customers queued at service counters and shook their fists. One might conclude that a load issue caused the failure (although an insider reported that it was the common promote-untested-code-to-production practice that caused the outage). Still, it provides an example where a business decision, a partnership ending, or other circumstance outside of IT’s control could cause a surge in load for which your gradient load tests don’t account.

Your Application Goes Viral

You probably already know the meaning of the Slashdot Effect, but I’ll explain it anyway: When the technology blog Slashdot² mentions a Web site or article, the resulting traffic often overwhelms underpowered Web sites. If everyone who reads

this magazine now goes to access the given link for the Slashdot website we are by association increasing their traffic. In the years since the Slashdot effect was first noted, other highly trafficked Web sites and social media, such as Facebook, Twitter, and Reddit, have arisen with the ability to focus a large number of users onto a Web site or application very suddenly.

Your organization might only discover the traffic spike when it happens, or when your application breaks because the thousands of new users arrived without stepping up gradually, patiently queuing in an orderly fashion to wait for applications, pools, or database connections to become available. Viral users are more like rowdy football fans, all scrambling for the entrance at once, than theatre patrons.

A Media Blitz

If your organization or application faces consumers instead of industry professionals, at some point the company might decide on a media blitz to drive consumer traffic, and this media planning might not take into account technical considerations. In my time at an interactive agency, we maintained a web site for a consumer packaged good complete with streaming media and data collection forms. Some months after we deployed the site, the client decided to promote the product through non-Internet means, including television commercials, print media, and press releases that gained the notice of national television programs.

Since these decisions lie outside of IT, sometimes they can be opaque or unnoticed. Unless you ensure your applications can handle that user assault at the beginning, you might be called upon to pick up the pieces very suddenly when your URL appears on the telly.

Don’t be Gentle

When you have a load testing tool at hand, you should use it not only to provide some idea of how your application will perform under load (the answer, regardless of the numbers, will probably be “good enough”). Instead, you should also use that load testing tool to try to break the software through testing the application with the maximum number of users all at once to see if your application can handle that meager number of users without choking on the various sorts of resource allocation issues that you would not find with a measured increase in test load. This is important because no matter what usage level your organization anticipates for your application, once you turn it on, you might find that the actual usage can suddenly and catastrophically spike beyond even your worst sleepless nightmares. □

REFERENCES

- http://articles.chicagotribune.com/2012-02-04/business/ct-biz-0204-walgreen-20120204_1_kermit-crawford-90-day-prescriptions-express-scripts
- What is the “Slashdot Effect?” - <http://slashdot.org/faq/slashmeta.shtml#sm600>



AUTHOR PROFILE - ADAM KNIGHT

Adam Knight is Director of QA and Support for RainStor inc. and writes an independent blog at <http://www.a-sisyphean-task.com>

Testing big data in an agile environment

By Adam Knight

One of the hot trends in technology this year is the topic of “Big Data” and products targeting the “Big Data” problem¹. In May 2011 McKinsey Global Institute described Big Data as “The next frontier for innovation, competition, and productivity”² and in December 2011 the International Institute of Analytics predicted that “The growth of “big data analytics” will outpace all other areas of analytics in 2012”³. Previously unseen levels of technology adoption across the world are producing greater and greater quantities of data which organisations are looking to analyse using scalable technologies such as the Hadoop distributed software architecture⁴. At the same time greater legislation necessitates the ability to store this information securely.

As big data presents fantastic opportunities to software vendors, it also poses some significant problems for test teams in small organisations who are working in this arena. The use of agile development processes is becoming ubiquitous in responsive organisations, with continuous integration and fast feedback loops allowing organisations to operate in responsive markets.

When it comes to testing long running, large scale big data technologies it can be very difficult to fit the testing operations within the short development iterations that typify agile processes. In my current organisation, our product is used to store many Terabytes of data that will typically take many servers months to import. In a staged testing development project we would expect to have a long running performance and load testing phase in which test to the performance limits of the system. How then, do we perform equivalent testing within

a 4 week agile iteration? In order to tackle this problem, we have to look at the software itself and test smarter, not harder.

Understanding the System

Big data tools by their very design will incorporate indexing and layers of abstraction from the data itself in order to efficiently process massive volumes of data in usable timescales. In order to test these applications our testing too, must look at these same indexes and abstraction layers and leverage corresponding tests at the appropriate layers. In this way we can test the scalability of the system components without necessarily having

to process the full data load that would normally accompany that scale of operation.

For example, within my current system data is stored in a database like structure of schema and tables. Data in each table is stored in 100,000 to 1,000,000 record-sized “partitions” of data. These partitions are then indexed via a metadata database which stores the appropriate metadata to look up the partition efficiently when querying the data from the application, as shown in Figure 1.

As testers we have to work to understand this metadata database and the relationships that exist between it and the data. This knowledge allows

Continued on page 9

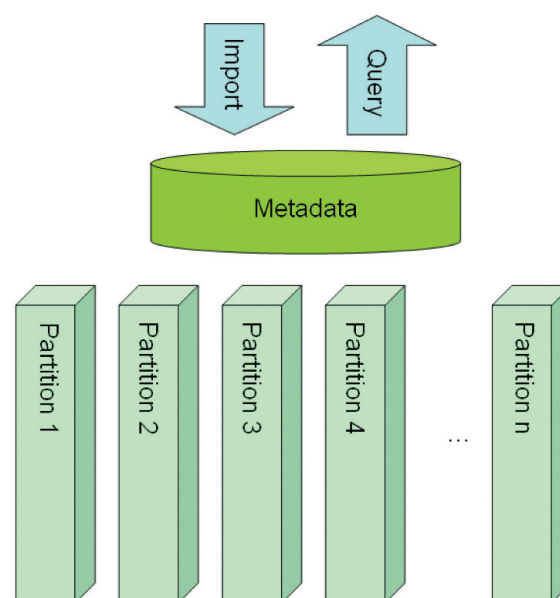


Figure 1. Metadata Layer Indexes Fully Populated Data Partitions

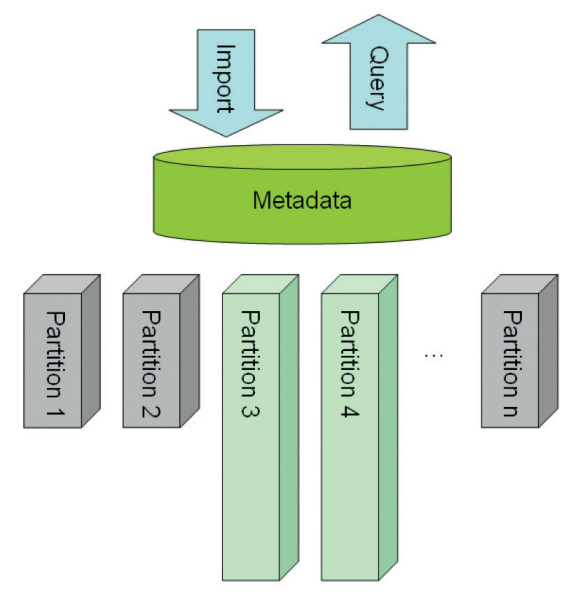


Figure 2. Fully Populated Metadata with Reduced Data Storage



Continued from page 8

us to create test archives in which each layer in the system behaves exactly as it would in a massive production system, but with a much lower setup and storage overhead (Figure 2). By essentially “mocking” out very small partitions for all but a target range of dates or imports, we create a metadata layer that is representative of a much larger system. Our knowledge of the query mechanism allows us to seed the remaining partitions (Figure 2 - Partitions 3 and 4) with realistic data to service a customer query across that data range that is functionally equivalent to the same query on a much larger system.

Hitting the Ground Running

Whilst it is important to have some testing of newly installed systems, many of the important performance tests that need to be performed on big data systems will be more realistic with a populated installation. Building each of these up from scratch can be a time consuming exercise which may not be possible within the confines of a sprint iteration. The use of virtual machines and the ability to ‘snapshot’ and roll back to a known hard disk state is a very useful tool for this kind of operation in smaller scale testing, but is of limited use for big-data archiving tests given the scale of storage involved. In order to overcome this problem, we can make use of various techniques to pre-populate data into the system prior to executing new tests. A few techniques that I currently use are:

Static Installation – Configuring the software against a static, “read-only” installation can be useful for testing query performance against a known data set for performance benchmarking.

Backup/Restore - Using the backup/restore and disaster recovery features of the system to restore an existing installation in a known state. As well as being an excellent way of restoring an installation this also helps to test the backup and recovery mechanisms themselves through real use.

Data Replication – If the software supports quick import or replication methods then we can leverage these to populate an installation with bulk data far more quickly than through the standard importing interfaces. For example, we utilise a product feature to support geographic replication of data across servers to bulk insert pre-built data into archives far more rapidly than the standard import process. Once we have reached a suitable capacity we can then switch to standard importing to test performance.

Rolling installation – Having an installation which is in a “rolling state” whereby tests import new data and archive out old data at a continuous rate. This allows for testing at a known capacity level in a realistic data lifecycle without the lead time of building up an archive for each iteration, with the additional benefit of boosting our version compatibility testing with an installation that has been running over a long period of time and many software versions.

Creative Testing

Big data is a growing market requiring specialist products, which in turn needs specialist testing. Performance testing as a devoted performance phase is no longer available to us when working with agile methodologies. To support our adoption of agile methods,

as testers we need to constantly use our creativity to find new ways of executing important performance and load tests to provide fast feedback within development iterations.

Here I’ve presented a few of the methods that we have used to try to address these challenges and support a successful big data product. This is certainly not a static list, as big data is getting bigger by the day. Even as I write this we face greater scalability challenges and increased use of cloud resources in order to ramp up the testing of our own Hadoop integration and consolidate our next step in the scalability ladder. As more companies look to move into the big data arena, I believe that testers are going to be a critical factor in the success of these organisations through their ability to devise innovative ways of testing massively scalable solutions with the resources available to them. □

REFERENCES

1. What is big data? Edd Dumbill, <http://radar.oreilly.com/2012/01/what-is-big-data.html>
2. Big Data: The next frontier for innovation, competition and productivity, McKinsey Global Institute, http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation
3. Analytics in 2012 Backs Big Data, Cloud Trends. Justin Kern, Information Management , <http://www.information-management.com/news/analytics-predictive-big-data-cloud-IIA-Davenport-10021670-1.html>
4. James Kobiels, Forrester “Hadoop: What Is It Good For? Absolutely . . . Something”, http://blogs.forrester.com/james_kobiels/11-06-06-hadoop_what_is_it_good_for_absolutely_something



Let's Test 2012

May 7 - 9
Runö Conference Centre
Åkersberga
Stockholm, Sweden

Michael Bolton Rob Sabourin
Scott Barber Julian Harty
Fiona Charles James Lyndsay
Rikard Edgren Henrik Andersson Leo Hepis

Michael Albrecht Chris Blain Anne-Marie Charrett
Selena Delesie Anders Dinsen Henrik Emilsson
Carsten Feilberg Markus Gärtner Dawn Haynes
Zeger van Hese Benjamin Kelly Simon Morley
Louise Perold Alan Richardson Torbjörn Ryber
Alexander Rotaru Huib Schoots Neil Thompson
Oliver Vilson Christin Wiedemann Johan Åtting

A European conference on context-driven software testing

The core mission of Let's Test is to help build an active community in Europe of software testers that either identify themselves with the context-driven school of software testing or are interested in learning more about it.
A full conference for only 15000 SEK. All inclusive!

www.lets-test.com

The periodic table of data – Pipe dream or possibility?

By Adrian Stokes

I work in a financial environment on a data warehouse creating business and management intelligence solutions in a mortgage and savings environment. As a

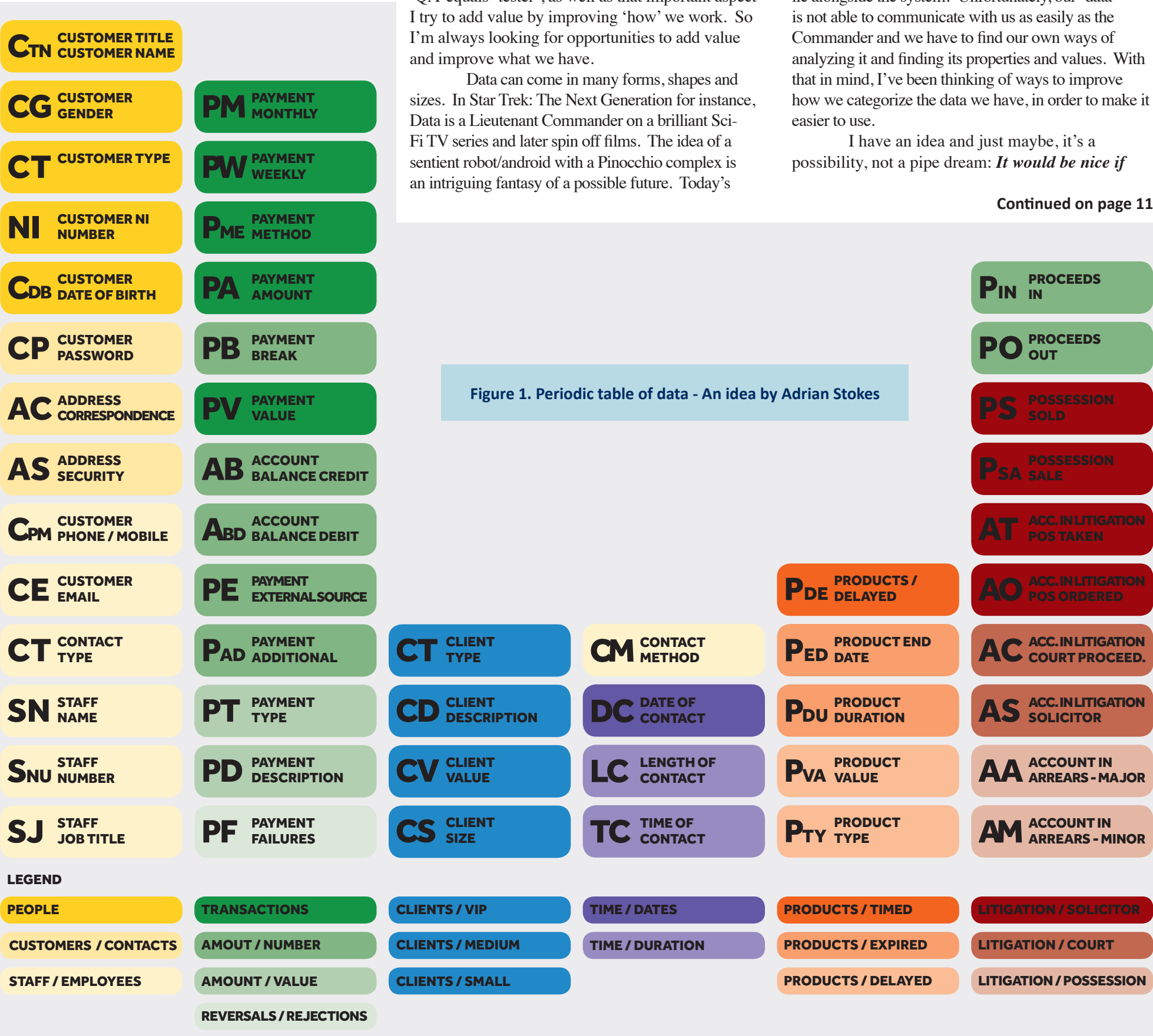
tester I don't want to just help deliver the best value I can to my team and the customer. I want to be a facilitator of learning and deliver value to myself and my company. My job title is Quality Assurance Analyst and while some have the perception that 'QA' equals 'tester', as well as that important aspect I try to add value by improving 'how' we work. So I'm always looking for opportunities to add value and improve what we have.

Data can come in many forms, shapes and sizes. In Star Trek: The Next Generation for instance, Data is a Lieutenant Commander on a brilliant Sci-Fi TV series and later spin off films. The idea of a sentient robot/android with a Pinocchio complex is an intriguing fantasy of a possible future. Today's

'Data' coming out of software systems, like Data after his 'emotion' chip can be so annoying! There's loads of it, everywhere you go, everywhere you look and every time you do something, more is generated. And that's before you even think about all the logs that lie alongside the system! Unfortunately, our 'data' is not able to communicate with us as easily as the Commander and we have to find our own ways of analyzing it and finding its properties and values. With that in mind, I've been thinking of ways to improve how we categorize the data we have, in order to make it easier to use.

I have an idea and just maybe, it's a possibility, not a pipe dream: *It would be nice if*

Continued on page 11



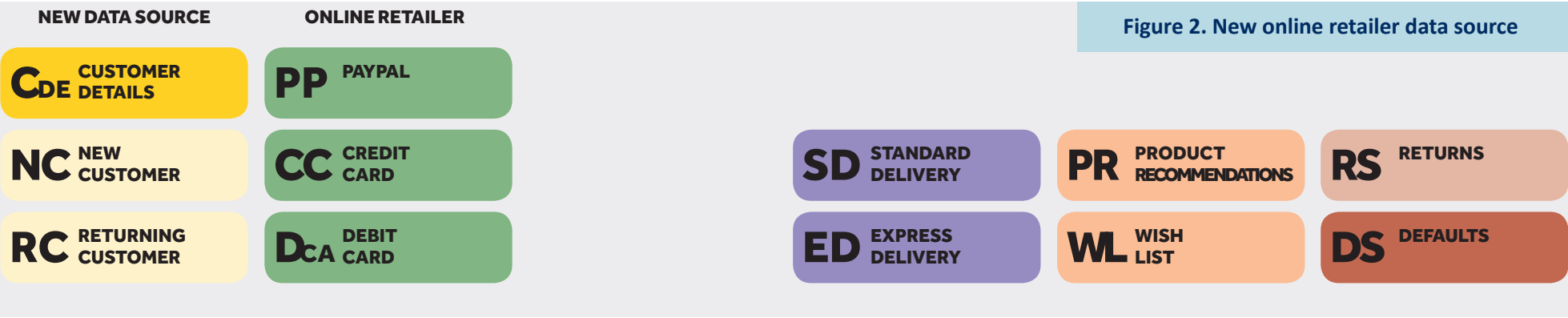


Figure 2. New online retailer data source

Continued from page 10

someone could come up with a periodic table of data or perhaps a set of them, for different subject areas!

I was told in school (and it's supported by Wikipedia¹ so it must be true...) One of the main values of the periodic table was being able to predict the properties of an element based on its location in the table. Unfortunately that's where my chemistry knowledge stalls but the principle of arranging your stakeholders, customers and users in a table still applies. Maybe it can also apply to your data! Imagine if you have a new data source to bring into your warehouse. How cool would it be to look at its properties and understand how it aligns with your existing data easily? To be able to know what areas you can enhance or expand into quickly before engaging in lots of analysis.

Figure 1 shows a very basic example for a mortgage data set that shows a few different areas that could be considered. I've color coded the different areas like 'people', 'payments' etc and used shades for the variations. I appreciate this is crude and needs work but that's where the testing community comes in. I'm sure there are many of you wiser than me who could improve this?

So now we have our premise, let's take some example of how it would work in the real world: You have been asked to bring in a new data source from a large online retailer. This new data source could have such data implications as customer details, contact numbers, tables for both

time of day and contact information, how they paid, if they wanted standard or express delivery. There could be a history that offers a chance to make recommendations or the customer may have defaulted on payment or had many returns.

In this new example (see figure 2) you can visually see that types of payments align with your transactions and payments. Customer details can be expanded and possibilities of how to use that new data alongside existing data and where they already match are quickly identified. Obviously this is just a small sample of the data you would see but hopefully it is enough to get the concept. A periodic table of your existing data like the first example could save days of analysis to identify new possibilities for using your new data shown above. Looking at the original and new data characteristics shows easily how these align and from there analysis is guided.

For those familiar with the world of data warehouses this could mean new data sets or extensions to dimensions or fact tables simply by seeing what fits where. It could align with your customer dimension to increase your effectiveness in communications. Confirm you are 'treating customers fairly'² by trending contacts via phone, post, email and letter. See how to best align your product offerings with potential customers.

Another example we could use is a travel company. As well as the standard customer details there could be favourite times and locations to consider. How many are in the party and what level of package they would prefer. The level of service they desire could link directly to the amount they pay so could align with your payments section to offer new ways of looking at customers. Perhaps those choosing half board would go all inclusive if given longer to pay?

Again you can visually see (see figure 3) how the new data lines up with your existing customer profile. Or if you are just starting to create your data table an example of another could help you decide

how you want to proceed and represent your data. It's like getting a new Lego set and working out if you can improve your existing models with it! Mapping the new data source against your existing table would expose what areas were affected and give a quick view of how the new data could enhance, improve or offer new opportunities for analysis. And once you understand the concept it should not take too long either!

A basic set of data could be common to most environments with perhaps a few tweaks. Customers, payments, clients and elements of time, be it payment schedules or just dates are familiar to nearly every project and product. Then variations on that common or generic set would apply like arrears or preferences etc. depending on what mattered to your business.

I believe there are all sorts of possibilities for the Periodic Table of Data's use depending on your environment. Any company that uses MI (Management Information) or data tracking to help make strategic decisions could make use of this. The main thing to remember is that this is an idea rather than a presentation of a technique or fully developed system. In offering it to the community my main hope is the readers will pick up the ball and run with it! If you think you can use this, improve this or make it work in your environment please do. I'd love to know how you get on. It may be I've gone into too much detail or not enough? It may need the areas or 'shades' redefining for your world? As I said, I'm sure the testing universe will have its thoughts and opinions and I'd love to hear them. My personal mail is adystokes@sky.com and I'd love to hear your thoughts.

Perhaps one day the Periodic Table of Data will seek out new paths and new learning's. It may boldly go where no visual aid has gone before! Or not? We will have to wait and see. □

REFERENCES

- 1. http://en.wikipedia.org/wiki/Periodic_table
- 2. <http://www.fsa.gov.uk/doing/regulated/tcf>

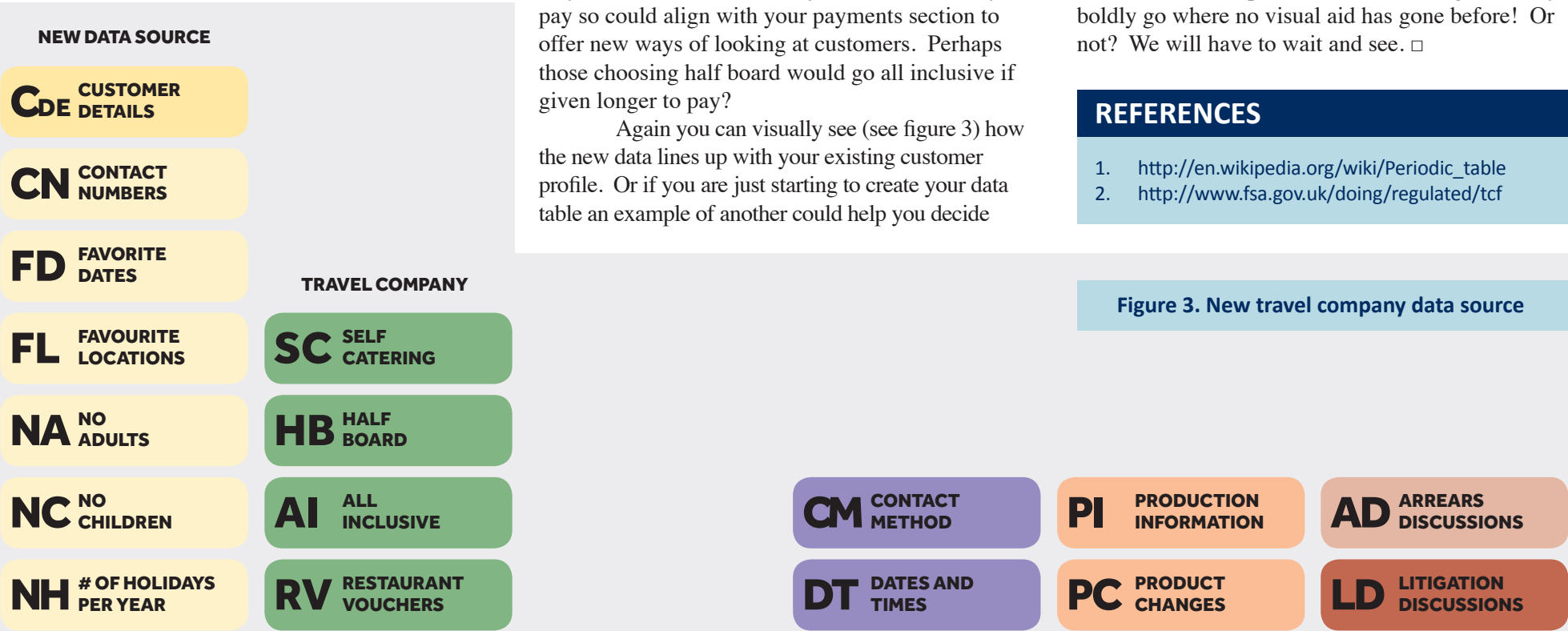


Figure 3. New travel company data source



A look inside... Sauce Labs

HELLO - WHO ARE SAUCE LABS AND WHAT DO YOU DO?

Sauce Labs is a small San Francisco-based company with deep technical roots in software development automation, security and high-performance infrastructure. We provide a cloud service infrastructure for running automatic Selenium tests or interactive exploratory testing. We currently give our users access to up to 150 virtual machines (VMs) in under 50 seconds and support 35+ OS/browser combinations.

I'M NOT REALLY FAMILIAR WITH "CLOUD BASED TESTING." CAN YOU EXPLAIN WHAT IT'S ALL ABOUT?

Software developers need to test their stuff before deploying it. It used to be they needed to conscript someone to the typically thankless task to requisition, install, configure, manage, update, patch and regularly restart the test infrastructure. As the pressure grew to complete slow and voluminous functional tests faster and faster to support accelerating release cycles, those labs had to grow and the headache of supporting them grew with it.

With cloud based testing that problem goes away. Through the power of the Selenium API and the connectivity of the Internet users can more or less instantly shut down their test lab machines and switch to a public cloud service. In the process, they enjoy more machines in parallel, more reliable test results, and access to new browsers and Selenium releases without having to trouble themselves with the unexpected but nevertheless usual yak-shaving¹ headaches that ensue.

For users whose staged application resides behind a firewall, there are a number of choices open to them for exposing that to the cloud service-hosted browser traffic:

- Move the app to a location outside the firewall with a hard-to-guess GUID-based URL.
- Add a login function to each script (I know, too painful to contemplate!)
- Work with a cloud testing provider who offers a two-way secure proxy for safe and secure through-firewall access as part of their service.

Some vendors offer such software, which serves one or more of three basic functions:

- Two-way secure proxy which connects the user's application to the cloud VMs.
- Intelligent content caching to mitigate the latency effect of testing over the Internet vs. locally (with an Internet in the middle, individual tests take longer, but the parallelism of the cloud overwhelms the latency setback).
- Isolate Selenium tests from underlying Internet transport issues by maintaining a redundant pool of TCP connections. When the cloud software notices a slow connect, it transparently switches to one of the spares from the pool before Selenium notices anything is wrong.

JUDGING BY YOUR WEBSITE³, IT LOOKS LIKE YOU WORK WITH SOME REALLY INTERESTING ORGANISATIONS. DO YOU HAVE ANY PARTICULAR SUCCESS STORIES?

We do. Our largest user is a major US-based airline that has bolted us into the automation system supporting their 200-strong Web development organization. They run a suite of tests Sauce as part of their check-in process so that they have a branch that's always green and ready to deploy.

Also, it's not widely publicized, but one of our biggest users is the Selenium project itself. By uploading pre-release versions of Selenium 2 to our cloud, they can test a broad array of Selenium functions against a reference website. We see this as a win-win for the community.

ANY NOT-SO SUCCESSFUL STORIES? WHAT KIND OF PROBLEMS DO PEOPLE ENCOUNTER WHEN MAKING THE LEAP TO THE CLOUD?

The biggest thing we run into is users who have inadvertently built local dependencies into their tests. Many users employ naive waits in their tests that get them into trouble when the latency of the Internet gets introduced to the equation. Of course having the Internet in the middle of a web application is much more real-world than laboratory-pure environments and investing the thought to make tests work in that context can lead to more meaningful results. We've written extensively about clever timing strategies that coach users to build tests that work robustly in real Internet environments.

OUR TEST ENVIRONMENT GOES DOWN A LOT. CAN YOU PROVIDE AN ENVIRONMENT TO US TOO?

We commit to an SLA for customers for 99.9% uptime. We beat that every month in 2011. Our goal is 99.99%. We're getting close to that. We haven't found a customer environment that comes close to that yet.

WE USE CLONE DATA IN OUR ENVIRONMENT - HOW CAN I BE SURE THAT USING YOUR SERVICE ISN'T GOING TO END UP IN A "MILLIONS OF CUSTOMER RECORDS STOLEN SHOCKER!" STYLE DAILY MAIL EXPOSÉ?

As you may know, the Selenium client server protocol runs over HTTP and effectively remote-controls a real browser to perform actions a real user would do. That browser accepts commands from the operator's end (the user's infrastructure) one command at a time, performs that command and reports back on what it was asked to find (was the user name on the page correctly, did the frequent flyer miles transfer go through, etc). Once each test completes, Sauce Labs destroys the VM. None of the browser cache data ever hits a spindle, it operates out of a solid state drive so that once a test finishes, absolutely no artifact remains: digital, physical or magnetic.

OK - LET'S SAY I'M CONVINCED. HOW AM I GOING TO MIGRATE ALL OF MY FUNCTIONAL/ACCEPTANCE TEST CASES INTO THE CLOUD?

Again, let's recall that Selenium is a client-server architecture and the way some cloud-hosted service providers have architected their service means users who adopt their service are moving the place they run their tests, not the tests themselves. For the typical user who is already kicking off their Selenium tests automatically, adopting the such a cloud service is as simple as pointing their tests CloudTestingProvider.com instead of LocalHost

Continued on page 13



Continued from page 12

and inserting a single JSON string in their unit test framework which specifies three things: what browser they want us to use, the URL of their staged application and their account key.

WHAT ABOUT PERFORMANCE TESTING?

A fair number of our customers encounter what we affectionately refer to as an “accidental load test” when they get overly optimistic about how many concurrent Selenium tests their staged environment can handle! All kidding aside, we see massive opportunity (read: we’re expanding as fast as we can to keep up) in the functional testing space. There are a number of offerings out there we’re acquainted with in the cloud-delivered load / performance testing space with established customer bases and with whom we enjoy good working relationships.

GREAT! NOW I’VE OUTSOURCED ALL OF MY TESTS TO THE CLOUD, I DON’T NEED TO HIRE TESTERS, RIGHT?

When I decided to rent vs. buy a car, I didn’t give up the need for walking. You still need to walk to and from the car regardless of whether you own and maintain it yourself. So it is with using the



cloud to run your tests. You still need people to do smart things. You just don’t need them to spend as much time doing the thankless things of setting up, configuring, restarting, upgrading, patching and generally being a slave to a test infrastructure.

Thanks to John Dunham (Sauce Labs CEO) and Ashley Wilson (Sauce Labs Customer Development Manager) for answering all our questions. □

REFERENCES

1. <http://catb.org/jargon/html/Y/yak-shaving.html>
2. <http://saucelabs.com/>
3. <http://saucelabs.com/blog/index.php/2011/04/how-to-lose-races-and-win-at-selenium/>



annual conference and exhibition test automation day

2nd edition, June 21st 2012, World Trade Center - Rotterdam, The Netherlands

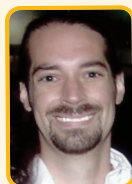
The Future of Test Automation

After the first successful edition of Test Automation Day (2011), we would like to welcome you to the 2012 edition in Rotterdam, the Netherlands!

The independent conference committee will organize an exciting collection of keynote sessions, business cases and workshops presented by national & international thought leaders and experts.



Elfriede Dustin



Scott Barber

International gurus as **Elfriede Dustin** (Author and Automated Testing Evangelist, Sr. Systems Engineer at IDT) and **Scott Barber** (CTO PerfTestPlus, Co-Founder of Workshop “On Performance and Reliability”) will contribute to the second edition! **Scott Barber will share his experiences during an exclusive (lunch) tutorial!**

Are you interested in this exclusive lunch? Register now! Only the first 50 registrations will be able to participate.

Register with the special member code!

As a reader of “Testing Planet”, you’ll get a **€ 100,- discount**.

Register with the code: **“TAD2012-TPL”** on www.testautomationday.com and pay only € 195,- (excl. VAT) for **Test Automation Day!**

MORE INFORMATION AND REGISTRATION AT WWW.TESTAUTOMATIONDAY.COM

€ 100,-
discount for readers of
Testing Planet
Pay only **€ 195,-**
excl. VAT

CONFERENCE ORGANIZATION

ckcseminars

FOUNDING PARTNER

SQUERIST

PARTNERS

IBM

MICRO FOCUS

PARASOFT
We make software work.

SPONSORS

InfoSupport
Solid Innovator

TREND
if the business process counts

TRICENTIS
Technology & Consulting

EXHIBITORS

KZA BV


SQS NEDERLAND B.V.

VX COMPANY

PARTICIPANTS KNOWN JANUARY 30TH 2012


FACT BOX MBTI ⁷	
First letter - Favourite world	Do you prefer to focus on the outer world or on your own inner world? This is called Extraversion (E) or Introversion (I).
Second letter - Information	Do you prefer to focus on the basic information you take in or do you prefer to interpret and add meaning? This is called Sensing (S) or Intuition (N).
Third letter - Decisions	When making decisions, do you prefer to first look at logic and consistency or first look at the people and special circumstances? This is called Thinking (T) or Feeling (F).
Fourth letter - Structure:	In dealing with the outside world, do you prefer to get things decided or do you prefer to stay open to new information and options? This is called Judging (J) or Perceiving (P).

FACT BOX BELBIN ⁸	
Plant	Generates ideas and solves difficult problems
Resource investigator	Explores opportunities and develops contacts
Co-ordinator	Delegates effectively
Shaper	Has the drive and courage to overcome obstacles
Monitor Evaluator	Sees options and judges accurately
Teamworker	Listens and averts friction
Implementer	Turns ideas into actions and organises work that needs to be done
Completer finisher	Polishes and perfects
Specialist	Provides knowledge and skills in rare supply




COP

STOP - This defect is a violation of the first requirement




DETECTIVE

Elementary my dear! The coffee stain here is our lead to the killer bug.




DUMMY

You want me to ram it into a firewall again? Yeah, sure!




EXPLORER

Hey, there is no path this way through the integrations – yet.




STUDENT

See folks, I learned something new today.



GUARD

This is my gate; you shall not pass without proper review.



HAZMAT

Gear up, we're deploying for the bug bash!

AUTHOR PROFILE - JESPER LINDHOLT OTTOSEN

Jesper Lindholt Ottosen is a senior test manager with CSC Denmark. He has 10 years experience of test management in large scale telecommunication enterprise environments. Jesper blogs internally in CSC on testing topics and was a guest blogger for EuroStar2010. Jesper tweets as @jlottosen. Jesper is an adult fan of LEGO and a senior test manager at CSC in Denmark. You can find him on Twitter and Google+ writing about software test and LEGO. This is fan art and not authorized by LEGO® or CSC.

REFERENCES

1. <http://blog.softwaretestingclub.com/2009/12/tester-types/>
2. <http://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/>
3. <http://www.readwriteweb.com/hack/2011/09/testing-your-code-with-myers-b.php>
4. <http://www.belbin.com>
5. <http://eurostarconferences.com/blog/2010/9/21/software-testing-is-a-skill-of-many-skills---jesper-ottosen.aspx>
6. <http://minifigures.LEGO.com>
7. http://en.wikipedia.org/wiki/Myers-Briggs_Type_Indicator
8. [http://www.belbin.com/content/page/5002/BELBIN\(uk\)-2011-TeamRoleSummaryDescriptions.pdf](http://www.belbin.com/content/page/5002/BELBIN(uk)-2011-TeamRoleSummaryDescriptions.pdf)

10

REASONS WHY AS SOON AS Y

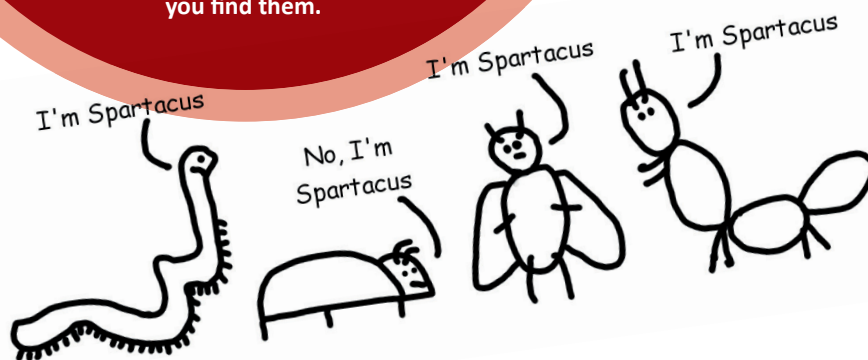
1. UNFIXED BUGS CAMOUFLAGE OTHER BUGS

How many times have you heard a tester say, "Good news, I've re-tested the bug you fixed and it's working perfectly, but I'm now observing a new bug"? You might be in luck, fixing a bug may reveal no further problems, but postponing this kind of discovery is a risky strategy. What happens if that Priority 3 bug you've been ignoring has been camouflaging a Priority 1, or worse still, a bug that will require significant alterations to the software to fix? If you have one of these bugs hiding somewhere in your code, the sooner you discover it, the better. Don't delay finding important problems; fix bugs as soon as you find them.



2. UNFIXED BUGS SUGGEST QUALITY ISN'T IMPORTANT

We're all professionals at heart, but it's surprising how quickly a team can find themselves in a downward spiral. A developer working on software that already contains hastily written, error prone functions, with little or no unit test coverage, is likely to add more code of the same nature. Similarly, a tester who has seen tens of reported bugs go unfixed is unlikely to be enthusiastic about reporting many more. Of course, it isn't just developers and testers that are affected. Over time, every member of the team will start to ask themselves, "what's the point", why aim for a high quality product when a substandard one is the accepted status quo. Don't set substandard quality expectations; fix bugs as soon as you find them.

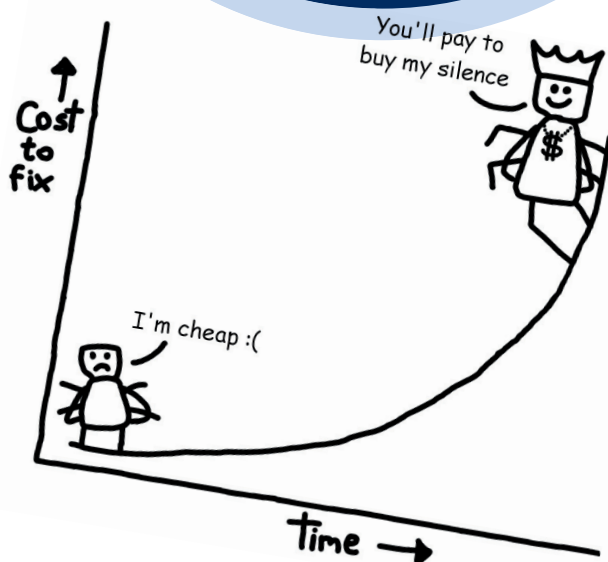


3. DISCUSSING BUGS IS A WA

Regardless of whether it's during a dedicated triage around a desk, discussing unfixed bugs is a waste of time. There really is only one question: "Should this bug need to be fixed"? Even if the answer is just noise. Should we categorize it as a Priority 4? How long will it take to fix? 113 first or bug 114? These are questions that should be avoided (including the often-asked "What's the follow-up") if teams fixed bugs as soon as they find them. Without doubt, every problem requires this level of attention. Don't waste time on unnecessary discussions; fix bugs as soon as you find them.

6. UNFIXED BUGS DISTRACT THE ENTIRE TEAM

When somebody encounters an unfixed bug a number of distracting questions are planted in their mind. Take a developer who is about to make an enhancement when they notice a bug. Should they fix the bug first, has somebody else fixed it but not checked-in, can they rely on the buggy code to be the basis for their own? Similarly, imagine a tester who has stumbled across a bug in one functional area whilst setting up the pre-conditions to test another. Should they postpone testing the intended area and instead explore around the bug they stumbled across, has this bug already been reported and would exploring it be a waste of time, could this bug (positively or negatively) pollute the results of the planned tests? Don't let your team be distracted by unfixed bugs; fix bugs as soon as you find them.



7. UNFIXED BUGS HINDER SHORT-NOTICE RELEASES

Once in a while an event occurs that forces a team to release all or part of their software when they least expect it. Maybe an emergency patch is needed to fix a bug in the production environment, or an unexpected visit from the project sponsor requires the latest release to be installed on a demo laptop. These events can be taxing at the best of times, often made worse by the presence of one or more unfixed bugs. It may only take a relatively short time to perform the release itself, but with unfixed bugs in the code, how long will it take to get the software ready for release? Even if a team can quickly fix any bugs blocking the release, there is also the time required to re-test the bugs to consider. The result is often a delayed release or a release that contains only the most glaring bugs removed. Don't let your releases be hindered by unfixed bugs; fix bugs as soon as you find them.



"Look at me! Aren't I a pretty little bug?!"

8. UNFIXED TO IN

No two bugs are the same. It takes seconds to investigate, minutes to reproduce, and minutes to fix. Combine together the time spent on unfixed bugs a project and you can see how easy it is to fall into a trap of re-test bugs fade away and they can't be fixed. Contingency – this is a place and detailed analysis of whether the contingency is worth the cost. Know how long it will take until the work is done. Stakeholders will be disappointed if the bugs are not fixed.

WHY YOU FIX BUGS YOU FIND THEM

3. UNFIXED BUGS LEAD TO WASTE OF TIME

As part of project planning, a meeting or just gathered to discuss fixed bugs is a waste of time. When that needs answering, "does anything else, whilst interesting, prioritise this bug as Priority 3 or take to fix? Should we fix bug or are all questions that could be in lengthy conversations that as soon as they found them. Project will have its share of attention, but few projects attention to be the norm. With unnecessary bugs as soon as and them.

What are you still doing on that chart? We were fixed last week!

Nobody told me!

No. of Bugs

Time

4. UNFIXED BUGS LEAD TO DUPLICATE EFFORT

The greater the number of unfixed bugs, the harder it is to identify whether a bug has already been reported. Imagine a scenario where there are only 5 unfixed bugs. When a "new" bug is discovered, it's easy to identify whether that bug has been reported by someone else. Now imagine trying to perform the same task when there are 50 unfixed bugs in circulation. It's either going to take a disagreeably long amount of time (time which could be better spent looking for other bugs) or the thought of such an overwhelming task will cause it to be abandoned, often leading to duplicate bugs being reported. These duplicate bugs lead to duplicate investigation and duplicate re-testing. Don't waste time on unnecessary duplication; fix bugs as soon as you find them.



6. UNFIXED BUGS LEAD TO INACCURATE ESTIMATES

Estimates are never the same. Some require mere minutes to investigate; others take hours to diagnose. Some can be fixed quickly; others take several days. Some can be automated; others require manual verification. These uncertainties and you can see why the more uncertain the estimates become. The trap of thinking that the effort required to fix bugs is into insignificance compared to other project tasks is rarely the case. Even with a contingency in the analysis of each bug performed to understand the contingency is sufficient, a team can never truly know how long it will take to fix and re-test each bug until the work is complete. Don't misinform your stakeholders with inaccurate estimates; fix bugs as soon as you find them.

STOP

You may not pass!

9. FIXING FAMILIAR CODE IS EASIER THAN UNFAMILIAR CODE

The human mind is capable of many incredible feats, but retaining information indefinitely is not one of them. Over time our memory decays and things we used to know intimately become blurred and unfamiliar. The code a team writes is no exception and for this reason it is easier for a team to fix a bug in code they edited earlier that day compared to code they haven't seen for a week or two. A team can reduce the effect of memory decay by sticking to good development principles, but this will only reduce the effect of memory decay, it can never alleviate it completely. Avoid the frustration caused by having to familiarise yourself with a piece of code you once knew; fix bugs as soon as you find them.

Did I really write this code?



5. UNFIXED BUGS LEAD TO UNRELIABLE METRICS

Different teams analyse bugs in different ways. Some casually monitor how many are left to be fixed, whilst others track everything from their density to their lifespan. Regardless of the complexity, every bug-based metric relies on accurate underlying data. As the number of unfixed bugs increases, it becomes increasingly difficult to maintain accurate bug information. Even if the information was correct at the time, the longer a bug is left unfixed, the greater the chance that information will diverge from reality. The resulting misinformation then ripples through the team. Don't fall foul of project decisions based on incorrect information; fix bugs as soon as you find them.

There's nothing to worry about. We "just" need to fix a couple of bugs and then we're done!



10. FIXING A BUG TODAY COSTS LESS THAN TOMORROW

For all the reasons listed in points 1 to 9, fixing a bug today will cost you less than fixing the same bug tomorrow. If a bug is left to fester in the software you are developing, configuring or maintaining it may camouflage other bugs, demotivate the team by suggesting quality isn't important, become the topic of pointless conversations, cause duplicate effort, lead to incorrect project metrics, distract the project team, hinder short-notice releases, invalidate estimates and lead to unnecessary frustration. And the longer you leave a bug before fixing it, the more likely these things are to occur and to a greater extent. Don't put your project at risk; fix bugs as soon as you find them.

What is static analysis?

By Chris Wysopal

Static analysis is a software testing technique that can be used to scrutinise all code paths and data flows that a program will execute without actually running the program. It does away with the need to build a potentially complex and expensive environment in order to analyze a software program for many classes of quality and security defects. Static analysis can also be performed early in the development lifecycle since it doesn't require a fully functioning program or test data.

It is useful to look at industries outside of information technology to understand how they use static analysis for testing. Modern jet engines require the highest levels of test rigor due to the immense cost of failure in the field ¹. Jet engine manufacturers have implemented testing throughout the manufacturing process. They don't wait for the complete engine to be built and fired up in a lab to do quality checks. They push the testing as early into the process as possible and as close to the component being built as possible ².

A titanium fan blade operates under immense stress as the jet engine fan turns at 10,000 RPM. A tiny fracture deep inside the blade may cause it to fail and break apart. The manufacturer could wait until the engine is fully assembled to test the blade by running the engine, but failure would be expensive and require extensive diagnosis to determine what actually went wrong. Instead, jet engine manufacturers use a static analysis approach. They perform an x-ray examination of the fan blades immediately after they are cast. If flaws exist they are identified at their source before the engine is completely built. Lengthy test execution cycles and corresponding triage of test-failures are not required. If cracks in fan blades cause failures look for cracks in fan blades!

The same approach can be used with software. We just need a technique for finding software defects that is as powerful as x-rays are for finding flaws in engine fan blades. We need to find the problems at their source, inside the code.

X-Rays for Your Code - Automated Security Flaw Detection

A Static Analyzer can have the methodology of the world's best security and quality code reviewers encoded into software to provide the depth of a manual code review with the accuracy, repeatability, and speed of automation ³. Static Analysis can be performed on either a program's source code or its binary executable. Both source code and executable will contain all of the semantics and logic that define the software's functionality. Binary analysis offers the convenience of not needing to have

access to all of the application source code in order to perform a complete analysis. With modern software construction techniques using off the shelf re-useable binary components and libraries this has an advantage. I am going to focus on using static analysis of binary executables to find security vulnerabilities.

The static analysis process starts with the binary executable of a program and builds a comprehensive and detailed model which represents what the CPU executes as it runs the program. The model is then analyzed by hundreds of application security scans that mirror the methodology of the world's best code reviewers.

Binary static analysis follows a 4 step process which starts with an application binary built for the target platform and ends with a detailed and prioritized report of the applications security flaws. The 4 main steps are outlined in Figure 1. Binary Static Analysis flowchart below.

The binary executable is the most accurate representation of how the program will execute in a given environment ⁴. The environment includes the operating system and libraries as well as the hardware architecture. Starting from the binary allows the model to include the transformations that the compiler makes when it generates the machine code. It also allows the static analyzer to look for vulnerabilities due to a programs interaction with components that are only available in binary form.

1. Loading

The analyzer loads the binaries into memory just like the operating systems would, enabling it to determine all of the interfaces that the program

makes to its environment through dynamic link libraries or required classes. These may be API calls to the operating system for services such as files I/O or networking or for 3rd party libraries the program uses such as a custom crypto library.

2. Application Modeler

Unlinking - The next step is to break up the monolithic binary into the individual functions that the programmers originally created. This is the opposite of the process that a linker performs when it takes the individual functions and stitches them together to create the entire program.

The result of the unlinking process is a call graph of the entire program and all the entry points. This call graph is used by the data flow modeler to graph the interprocedural data flows contained in the program.

Data Flow - Perhaps the most important dimension of any application model to be used for security analysis is the data flow graph. At the macro level, data flow modeling is an important part of doing a threat analysis of software because supplying specifically crafted data is the only way that an attacker can interact with an application to exploit security flaws. At the micro level, following data flows accurately is critical in determining whether a line of code is correct, risky, or a serious error. The data flow graph is used extensively by the security analysis scans to perform risk analysis on potentially dangerous coding constructs.

Before the data flow can be graphed the variables need to be discovered through a process called "variablization". The analyzer starts by

Continued on page 19

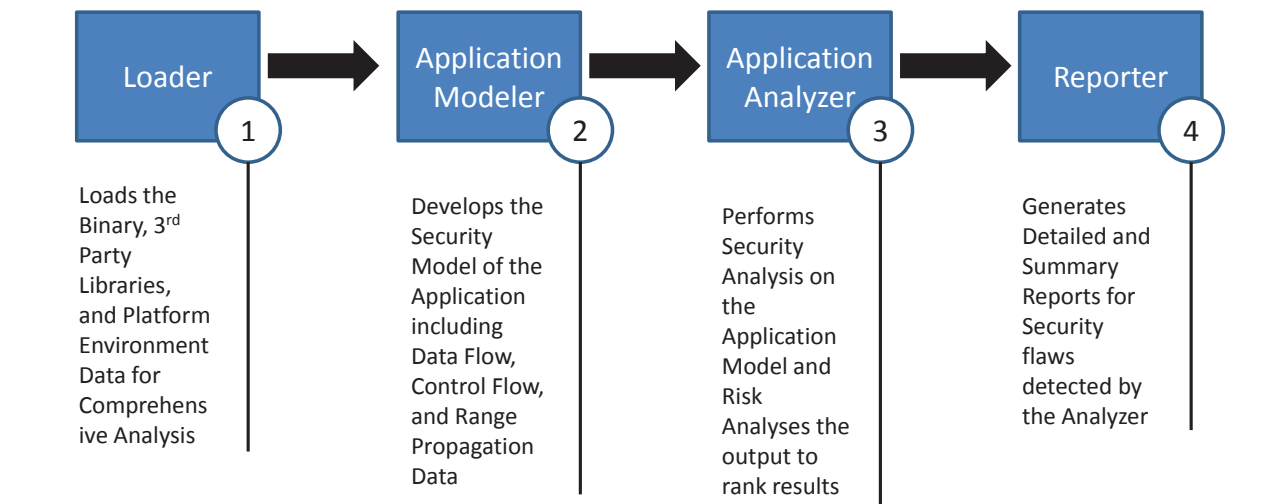


Figure 1. Binary Static Analysis flowchart



A Tester is someone who destroys the illusion that the application works. - by Gergely Brautigam

AUTHOR PROFILE - CHRIS WYSOPAL

Chris Wysopal, co-founder and chief technology officer of Veracode, is responsible for the security analysis capabilities of Veracode technology. He has given keynotes at computer security events and has testified on Capitol Hill on the subjects of government computer security and how vulnerabilities are discovered in software. He also has spoken as the keynote at West Point, to the Defense Information Systems Agency (DISA) and before the International Financial Futures and Options Exchange in London.

Continued from page 18

understanding the assembly language instructions that the program is compiled to for a particular hardware platform. It analyzes the instructions that manipulate data, moving it between the CPU registers and memory. From these instructions the registers are unpainted ⁵ back into the variables that the programmer specified and the types of these variables are discovered. For each procedure in the program, the input, output, and local variables are discovered and placed into the data flow graph.

Once the variables are discovered the data flow can be mapped as variables are initialized and data is moved into and out of them.

Control Flow - Determining the control flow within a function in the program allows the analyzer to discover the high level language constructs that were used by the programmer. This is where if-then statements and for and while loops are determined. Later in the security scans process the control flow graph will be used to determine whether or not return values were checked before use. The control flow graph will be an important input into the range propagation process.

A control flow graph (see main image) is generated by analyzing the branching instructions contained in the assembly language instructions with a function. The function is broken down into basic blocks, which are lines of code that are executed in sequence, and conditionals such as a jz (jump if zero). By knowing the way the compiler uses conditionals to compile if-then statements and loops, the analyzer can recreate a functionally equivalent high level representation of the control flow that the programmer intended. If this was represented graphically is would look like a classic flow chart.

Range Propagation - The range of a variable's potential value at any point in the code gives

important information that can be used for security analysis. If there is a conditional such as:

```
if (n < 100)
    strncpy(dest, src, n)
```

The range of n is between 0 and 99 (assuming n is an unsigned integer) within the basic block of the true clause of the conditional. This information can be used by the security scans to analyze whether a security problem such as a buffer overflow can occur at a particular line of code. By knowing that the value of n in the example above must be lower than 99 the scan can determine if the destination buffer, dest, is large enough to fit the potential range of sizes of the source buffer, src.

3. Application Analyzer

The essence of secure programming is to understand the limitations of the particular programming environment a program is built in and how to stay within the bounds of security correctness in this environment. The security scans are designed to detect when a programmer has exceeded the bounds of security correctness. After the analyzer builds a detailed model of the program the next step is to use that model to perform a security analysis. The security analysis is made up of a set of hundreds of scans that identify potential flaws, determine whether the potential flaw is an actual flaw and then determine a risk level associated with each actual flaw. The detailed model that the analyzer builds enables the security scans to have enough information to accurately rank the risk of a potential flaw as well as minimize the number of flaws that require some manual analysis to determine risk.

Triggers and Risk Analysis - The security scans start by searching for triggers in the applications model. Triggers are known problematic procedure calls such as the printf family of functions in the C standard library or security critical operating system APIs. Once the scan has found a trigger it investigates the context surrounding the trigger in the data flow, control flow, and range propagation models. This is the risk analysis portion of the security scan. The risk analysis will determine if the trigger has pointed to a security flaw or not. If it has, it will rank its severity. The risk analysis may also establish that there is not enough information in the application model to determine whether or not a trigger has uncovered a flaw or not. It will, in that case, flag the line of code as a possible error and offer information to help in the manual investigation of that area of code. Finally, the risk analysis may determine that there is no flaw and move on to the next trigger. The risk analysis functions are able to use the depth of the application model to bring the amount of manual analysis to an absolute minimum.

Types of Scans - A typical static analyzer will contain triggers and associated risk analysis functions in 5 main categories:

- Data injection flaws: SQL injection, command injection, cross site scripting (XSS)
- Memory corruption: Stack/Heap buffer overruns
- Information leakage
- Integer overflows/underflows
- Threading/race conditions

These categories of flaws account for the majority of critical security vulnerabilities. These flaws are the root causes that lead to exploits such as remote execution of arbitrary code, privilege escalation, and denial of service. For each scan category, the analyzer has rules created from secure coding guidelines and real world instances of software security vulnerabilities.

Here is an example of a trigger and risk analysis in action. This code comes from an open source project, tinyhttpd release 0.1.0 ⁶. Some of the code has been removed for brevity.

Continued on page 20

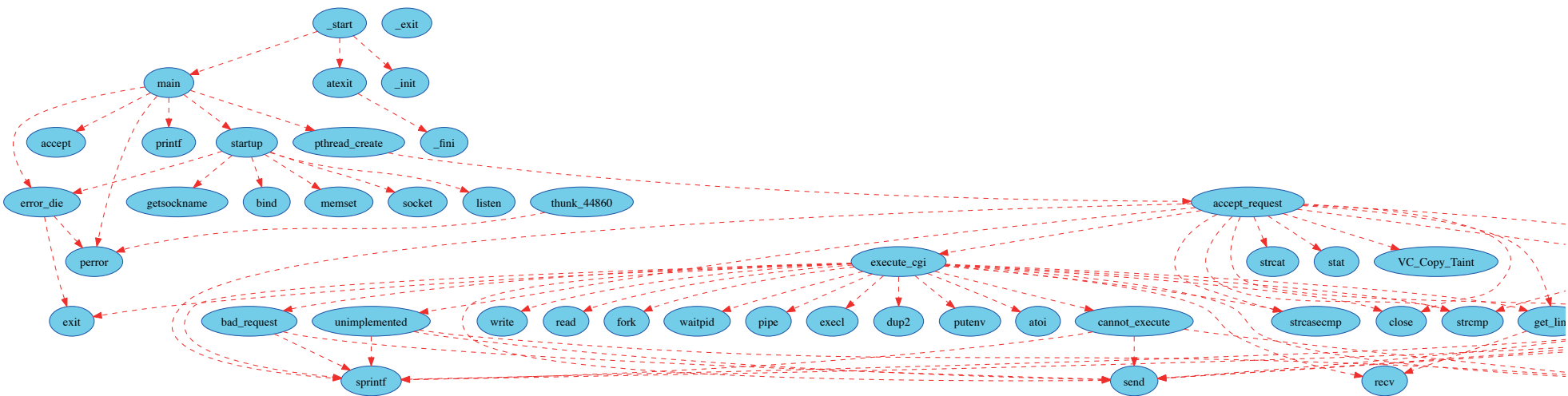


Figure 2. Control Flow Graph - For full size image visit - <http://bit.ly/wT7z5s>



Continued from page 19

```
void execute_cgi(int client, const
char *path, const char *method,
const char *query_string)
{
    char query_env[255];
    sprintf(query_env, "QUERY_
STRING=%s", query_string);
    return;
}
```

The trigger in this example is the sprintf function call that contains a %s format specifier with no precision specifier (i.e. %.100s for 100 characters max). The trigger causes the static analyzer to probe deeper into the control flow and data flow graphs of the program to determine the risk level of the code.

First a little background on sprintf secure coding issues. The sprintf function ⁷ can take 3 or more parameters. First is the target buffer, second is the format string specifier, then a variable number of source buffers. In the example above there is just 1 source buffer. The sprintf function assumes an unlimited size target buffer. The precision specifier is an integer positioned in front of the 's' in the format string to specify maximum string length. The sprintf function starts by copying the format string character by character. When it reaches a %s it copies the data in the first source buffer into its place. When there is no precision specifier the sprintf function will perform a memory copy of the data from the source buffer to the target buffer. It won't stop until it reaches a NULL terminator in the source buffer.

Now here is the security issue. If the source buffer size plus the format string size (minus 2 characters for the %s) is larger than the target buffer there will be a buffer overrun. The risk analysis function must determine the size of the source buffer and target buffer by performing data flow analysis. Once it has these sizes it is simple math to compute whether or not there is a potential buffer overrun.

The size of the target buffer was easy to determine. The variable, query_env, is a local static buffer of size 255. The source buffer is different. It is passed into the function as a pointer to a buffer as an input parameter, query_string. So to determine its length we need to perform interprocedural data flow analysis. This means we look for all the places in the code where the function containing our trigger is called and perform a data flow analysis to determine the size of the buffer passed into that function. In this particular case there is one place where the function is called. It is at the end of the following function, accept_request(). Some code has been removed for brevity.

```
void accept_request(int client)
{
    char buf[1024];
    int numchars;
    char method[255];
```

```
char url[255];
char path[512];
size_t i, j;
int cgi = 0; /* becomes true
if server decides this is a CGI
* program */
char *query_string = NULL;

numchars = get_line(client, buf,
sizeof(buf));
i = 0; j = 0;
while (!ISspace(buf[j]) && (i <
sizeof(method) - 1))
{
    method[i] = buf[j];
    i++; j++;
}
method[i] = '\0';

i = 0;
while (ISspace(buf[j]) && (j <
sizeof(buf)))
    j++;
while (!ISspace(buf[j]) &&
(i < sizeof(url) - 1) && (j <
sizeof(buf)))
{
    url[i] = buf[j];
    i++; j++;
}
url[i] = '\0';

if (strcasecmp(method, "GET") ==
0)
{
    query_string = url;
    while ((*query_string != '?') &&
(*query_string != '\0'))
        query_string++;
    if (*query_string == '?')
    {
        cgi = 1;
        *query_string = '\0';
        query_string++;
    }
}
execute_cgi(client, path,
method, query_string);
close(client);
}
```

The dataflow graph continues up through the function call to execute_cgi(). The parameter in question is the last one, query_string. The variable query_string is assigned from the variable, url, which is a local static buffer of size 255. Now the source buffer size is determined. Returning to the trigger source code, the size calculation using our buffer sizes can be performed to determine risk.

```
sprintf(query_env, "QUERY_
STRING=%s", query_string);
```

Source length + format string – 2 (for the %s) is

268. Target length is 255. The target buffer is not big enough. If an attacker crafts a query string with a length greater than 242 there will be a buffer overrun and the likelihood that the attacker manipulating the size of url could execute arbitrary code. This is a severe error and would be marked as such by the analyzer.

It is interesting to note that the tinyhttpd programmer did a good job of reading data from the network with a hard size limit and then parsed the data safely into a few buffers. But when the data was passed to another function and the subtlety of the sprintf function was involved, a security flaw was born.

With its comprehensive set of triggers and risk analysis functions, built on top of a deep model of the program, static analysis can uncover security flaws that many expert code reviewers would miss. And unlike manual review, it can be performed consistently and with high speed.

3. Reporter

Static analysis typically produces 2 types of reports for different audiences. The detailed reports are designed for the developer who is assigned to remediate the security flaws. The summary reports are designed for QA or management who are tasked with tracking security quality from a project or organization perspective.

Summary Report - At its simplest, the summary report boils the security flaws down to one number, the security score. This number is computed by the number and severity of all the security flaws in the application being tested. It gives an overall security quality metric that can be used to track security remediation of an application or perhaps acceptance test one. It can also be used to compare the relative risk of the different applications developed and used within an organization. The summary report also contains charts of the security analysis results by severity and by type, sorted by risk.

Detailed Report - A detailed report is the core output of static analysis findings. This is the information that points to the exact line of code where the security flaw is and gives details explaining what is wrong with the code and suggestions for how to fix it. In the sprintf() example given previously, the detailed report may recommend that the programmer use a precision specifier to limit the size of the source buffer that is copied to the target buffer.

Workflow - Static analysis tools and services are designed to fit the workflow of either a centralized application security group responsible for the security of many applications across an organization or a software development team building a single application. The workflow consists of 4 steps.

- Prepare a build of the software
- Perform binary analysis and run security scans to generate results
- Perform additional manual analysis triage on results (optional)
- View results and remediate.

Continued on page 21



A tester is someone who, says<it depends on the context> - by Jesper Lindholt Ottosen

Continued from page 20

This workflow can be integrated into any development process and can start as early as the software is coded and compiled. If testers can test the software or if a developer can debug it then it is ready for static analysis to automatically find security flaws in it.

Detect and Remediate Security Problems at the Source - Static analysis detects application security vulnerabilities at the point of their creation, insecure code. It then guides the remediation of vulnerabilities to the exact lines of code that caused them. It can do this quickly, precisely, and consistently. It assures you that your applications are coded following modern secure coding practices.



Manufacturers of modern mechanical or electronic technology don't let their customers see failures before looking for flaws. They don't even wait until the product is fully assembled. Once a particular class of flaws is well known it is inspected for during the production cycle. Static Analysis brings this level of repeatable, consistent quality to securing the software development cycle. Welcome to the new era of software testing and software security. □

REFERENCES

1.

<http://www.oobject.com/category/jet-engine-test-videos/> <http://cp.literature.agilent.com/litweb/pdf/5964-0156E.pdf>

2.

<http://cp.literature.agilent.com/litweb/pdf/5964-0156E.pdf>

3.

<http://www.slideshare.net/denimgroup/static-analysis-techniques-for-testing-application-security-houston-tech-fest>

4.

<http://research.cs.wisc.edu/wpis/papers/wysinwyx05.pdf>

5.

The term unpainting comes from the compiler term, register painting, which is a task a compiler performs when mapping variables to registers through the process of register allocation.

6.

Source is available at tinyhttpd.sourceforge.net

7.

<http://www.cplusplus.com/reference/clibrary/cstdio/sprintf/>



Crossing the testing minefield

Let me take you on a journey ...

By Mike Talks

You are with a party of friends and on a beautiful walk in the countryside. It's a lovely day, and according to your map you are making good time. Sometime soon you should arrive at your destination - a cafe which has come highly recommended to you. You can almost smell the coffee and cakes now!

As you open a gate you accidentally let a sheep through, which bounds off in front of you. Imagine the total shock when with a sudden BOOM it explodes before your eyes. It's at that moment you realise that you've somehow entered a minefield.

Suddenly your beautiful day has become

treacherous and your good progress has led only into peril. The only way you'll get out alive is by making good decisions from here on in.

There is no doubt that software development echoes this parable in many ways, especially when it comes to testing. Our projects can often seem to make good progress, but little do we realise we're just getting ourselves deeper into a minefield.

As testers we can often feel much like that rambler, we know there's peril out there in the form of defects, but we're not sure where exactly and of what severity. But often to the rest of our party the wake-up call will only come when the first mine has gone off ...

How do we best deal with minefields? And

how can we apply that to software?

One approach would be just to back up and hammer a sign (okay, being very careful where we hammered it) to warn "Caution Minefield". Then we can quite happily state that anyone entering does so at their own risk, and we did warn them. Some companies have indeed taken this attitude with customers, saying "well our customers will let us know if they find any problems". The problem with letting customers loose in a minefield is that you soon find yourself running short of customers ...

In a similar vein, we can follow the example of history, and drive a flock of sheep over

Continued on page 22



Continued from page 21

the field, and find problems that way (helps to have a bottle of mint sauce handy though). This can be what happens when we just try and let loose end users on our system in a controlled way, and “if it doesn’t fall over it must be okay,,, And nobody will be harmed”. This is rather quick and dirty way to test, and although it should find a lot of problems, it’s rather random, and not guaranteed to make it completely safe. Who knows what bits the sheep might be avoiding... Indeed the sheep may have a mentality towards self-preservation the shepherd isn’t aware of.

Much better is to bring specialist mine hunters or engineers in to clear a path. They will need some expensive equipment like metal detectors to probe ahead, and when a mine is found, they need to work with an expert to remove and defuse it. But this is slow and methodical work, with everything stopping when a problem is found. In the story above, as your engineers clear a path, there will be someone who will complain it’s too slow and the cafe will be closed by the time you’re across the field.

And though you’ll have cleared a path, the mine may still not be safe. Your friend needs to go to the toilet behind a bush whilst this is going on, but that involves stepping off the safe path. When can you declare a field safe and move on? Do you keep going backwards and forward with your engineers until you’ve covered every square inch? Do you cover the main footpaths first? Or can you claim it safe if you’ve not found any mine for the last couple of hours?

How can you ever really be sure? It’s easy to know about the presence of mines – they have a way of making their presence known - but how can you be confident of their absence?

That question is of course one that applies to testers, managers, developers. How can you be sure you’ve mine-cleared your software of defects? Of course you never can be sure, but it doesn’t hurt to have covered as much ground as possible. Every landmine you find makes the field safer.

In reality we tend to bring in our specialist demolition engineers, our testers in to check methodically as many paths as possible. Then to cover ourselves, we bring in our users for acceptance, and send out the sheep ...

There’s no ideal way to really cover off this scenario. No matter what the risk, there is always a drive to get to the cafe. But at the end of the day, the most important thing is we and everyone in our party respects that we’re in a minefield. And not to let our leader persuade us “you’re all being silly, it’s just one mine, c’mon follow me, they’ll be out of meringues if we dawdle”.

AUTHOR PROFILE - MIKE TALKS

Mike Talks is a software tester with Ki-wibank who’s based in Wellington. He’s working on an electronic book, but his biggest achievement thus far has been crashing the conversion application repeatedly. This it seems is the tester’s lot in life, we only get the broken toys ... ;-) He’d like to thank his son Cameron for his knowledge of military history in writing this article.



Can we say “no”?

By Markus Gärtner

Can we say “No”? Of course we can. We just have to press our teeth on our lower lip to form an “n”, followed by opening our mouth and speaking out the vowel “o”. “No.” Obviously anyone in the business of software development is capable of saying “no”, regardless of how impaired your co-workers might be. So the question rather becomes a matter of when to say “no.”

When to say “no”

In his book *More Secrets of Consulting*¹ Jerry Weinberg describes that he has a medallion as part of his consultant’s self-esteem toolkit. This medallion has “yes” written on one side, and “no” written on the other side. He describes how this medallion reminds him not only to say “yes” and to say “no”, but also to really mean the word being said. The point of the lesson being that we should say “no” when we really mean “no”, and conversely that we should say “yes” when we really mean “yes”, rather than “no”.

But how do we find out when we should say “no” and really mean it? It seems there are some occasions in which we can find out whether we meant “yes” or “no” just after the fact. Most often these are the occasions when we accuse our problems of lying in the very nature of software itself. However software by its definition² does not solve, nor is it responsible for any problem. Rather,

it may be us who are both having, and being the source of the problem.

For example, consider a programmer who is working overtime during a critical phase of their team’s project. For them it seems as if the requirements from the customer keep on changing all the time. What is the problem? Who has the problem? And who can solve the problem?

The problem may seem to be that the customer is unclear about what they want. Or perhaps that the customer cannot express their requirements well. It may also be that the requirements need to be adjusted over time to reflect changing market conditions, or a competitor innovation causes a change in the perception of the software under development.

Instead of the ever changing requirements and change requests coming from the customer, maybe the project manager is the problem. Unable to perceive that the project is already bloated with feature requests, the project manager may say “yes” to each and every request for change coming from the customer, not understanding that the project has already consumed all of its slack with everyone working at their limits. The project manager can solve this problem easily by saying “no” to any change requests that the team cannot achieve within a reasonable timeframe. At the very least they should check back with their team on this before saying “yes” to the customer or stakeholder. Of

Continued on page 23



A tester is someone who, after you’ve taken out the trash, shows you that the rest of the house is also in need of cleaning!” by Ola Hyltén

Continued from page 22

course, the project manager needs transparency and insight into the project deliverables in order to make good decisions in this regard.

Even if the project manager does not confirm the project schedule with their team, the team are still capable of giving the project manager feedback. As we saw earlier, every programmer and every tester should be able to tell their project managers “no” on their requests. Either by speaking up clearly, or by writing the letters down on a piece of paper, or even in an email. When the individual programmer or tester knows that he needs to sacrifice technical proficiency or personal time or even both to the new requests, they have the legitimate right to say “no” and really mean it.

For example when the project manager asks the team of testers on the project to work weekend overtime for eight weeks straight, the test-lead or manager as well as every other tester on the team should be able to say “no”, if they believe the demand is unreasonable. At which point, the project team may take a step back and seek to identify the underlying problem (of which excessive workload is a symptom), thereby fixing the development process.

Another example is the project manager that ships the product without running all reasonable tests, either by excluding some of the available regression tests, or by omitting unit testing completely. There may be good reasons to do this, but the team that is not responding with a clear “no” to these decisions is creating their own technical problems resulting from untested or complicated code, or hidden issues with the resulting bugs coming back from the production environment. With a clear “no” the project team makes their viewpoint clear, that such a decision is probably introducing even more problems in the near future than the team is willing to take on.

Finally, we may find ourselves in a company or culture where saying “no” is not appreciated or simply not allowed. For example, a tester communicating bad news all the time may find themselves labelled a “nay-sayer” or not unsupportive of the project. In such circumstances, the tester may be fortunate in finding a polite way to express their disagreement. In rare cases where this is not possible they may find the not so polite final way to say “no” by resigning from the team or the organisation. Though this may sound radical, sometimes there is little other alternative left.

How to say “no”

Unfortunately just saying “no” when you mean it may not be sufficient for your “no” to be heard. In order to make a clear statement on our individual position, we need to know how to say it in such a way that it is clearly received and understood. Since a simple “no” may not reveal which part of the request we are saying “no” to, we need to elaborate on our reasoning for saying “no”.

It may be hard for the receiver of the message to understand whether we are saying “no” to them personally or whether we are saying “no” to the content of the request. In the former case, the individual may feel offended. They may decide to put us in the “nay-sayer” or “unhelpful”

AUTHOR PROFILE - MARKUS GÄRTNER

Markus Gärtner works as an Agile tester, trainer, coach and consultant with it-agile GmbH, Hamburg, Germany. Markus founded the German Agile Testing and Exploratory workshop in 2011, is one of the founders of the European chapter in Weekend Testing, a black-belt instructor in the Miagi-Do school of Software Testing, contributes to the Agile Alliance FTT-Patterns writing community as well as the Software Craftsmanship movement. Markus regularly presents at Agile and testing conferences all over the globe, as well as dedicating himself to writing about testing, foremost in an Agile context.

mental pigeon-hole. So, we need to deliver the “no” message clearly, as an opposition to the content of the request rather than an opposition to the person.

Family therapist Virginia Satir developed a model³ for congruent communication that helps us understand how to deliver our message appropriately on both a verbal and non-verbal level. In her model there are three parts that need to be considered for congruent communication: the self, the other, and the context:

- 1. The self is the opinion of the speaker.
- 2. The other is the position of the communication partner.
- 3. The context is the thing that the communication is all about.

When we examine our simple “no” from the perspective of the Satir model it becomes evident that the message contains only the opinion of the self and will lead to incongruent interactions. I suggest instead that we communicate in a manner that incorporates all three layers of the Satir model.

We need to somehow deliver a message that appreciates the decision that the other has asked of us, stating for example that “I am grateful that you come to me with your request.” This delivers clearly the respect that we have for the position of the messenger. We appreciate that we get asked for our professional help. Making this appreciation explicit shows our gratitude for having been respected as the technical experts and reaffirms the position of the communication partner.

This still leaves out the contextual position however. We need to make clear to which part of the request we are opposing, maybe accompanied with a simple statement on our position. So, a congruent

response for the request to work overtime on the eighth consecutive weekend could be “I appreciate that you have come to me with your question. But I’m too tired to work overtime for the eighth weekend straight.” Or the question to ship the software without finishing off all the tests may be responded with “I appreciate your concern that we may lose market share if we don’t ship right now. But since my name will be on these classes, I want to make sure we deliver high quality work. To me this includes good unit testing.” By responding with the context in mind, the interaction is going to be much more productive.

Neither are spoken words the only way to say “no”. We should aim to deliver the message using appropriate body language⁴. Delivering the message that we are deeply sorry to reject the request accompanied in behaviour as well as in our words, helps the other party to understand our message. For example, the tester rejecting the question to work on the weekend may show his mood in a facial expression, and the developer continuing to write unit tests may make a regretful gesture with her hand.

Just do it

At times, it may seem hard to say “no”. You have to make yourself aware that it may be better to say “no” in a clear manner today than to say “maybe” now and “I told you it wouldn’t work” tomorrow. In either case, make up your mind, and say “no” when you mean it. You can practice saying “no” to many everyday things. For example try some of the following exercises saying “no” to checking your emails every five seconds, or to browsing the web instead of working on the production class, or to reading the latest article instead of writing the test report your boss asked you for. But you can also say “no” to your boss or project manager, when you really think that his request is going to guide you to the path of the next project death march.

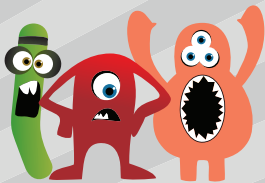
So, can you say “no”? □

REFERENCES

- 1. Gerald M. Weinberg, More Secrets of Consulting, Dorset House, 2001
- 2. <http://www.merriam-webster.com/dictionary/software>
- 3. <http://www.springerlink.com/content/18yvhtk2bw6wqt1n/>
- 4. <http://www.amazon.co.uk/Definitive-Book-Body-Language-Attitudes/dp/0752858785/>

WWW.SOFTWARETESTINGCLUB.COM

Software Testing Club
a community for software testers



A Tester is someone who finds a misspelled error message, red flags the distribution, and calls it a day. - by Sean P. Morley



The Evil Tester Question Time

Provocative advice for testers who don't know what to do!

Q1. DEAR EVIL TESTER...

What's all this I hear about how developers should do eunuch testing? - James

Dear James,

I can't tell if it is your hearing, or your reading, that is at fault.

If it is your hearing then you have misheard the word "unit". Unit testing is where developers test an individual thing. For example, if I was a developer and I had just written a system, then that system is an individual thing. So I (as a developer) have to write what is known as a "Unit" test, i.e. an individual test. So I write a single test to cover my entire system as a Unit.

This has now evolved in Agile projects into an advanced process called TDD or Test Driven Development.

In TDD testers are given complete control over the project and drive it to completion, which they do by shouting at the developers as they sprint around the room. They typically shout things like "Now write a failing test", "Check your style", "Re factor it into your poker" and other suitably motivating slogans.

If it is your reading at fault then you probably mean "enough" testing. And as we know,

there is no such thing as "enough" testing.

And how do we know? Because asking "How much testing is enough?" is like asking "How long is a piece of string?"

As an example, I have a piece of string on my desk, it is 302 mm long. And it isn't long enough. Hope this helps you in your next stand up meeting,

UNCLE E

Q2. DEAR EVIL TESTER...

Are you truly evil or just misunderstood? - Vernon

Dear Vernon,

Probably neither. In an attempt to make this section more interactive I offer you more options, could it be:

1. I'm deliberately engaging in false advertising for shock effect.
2. I can't spell. I meant to write "live" tester, incorrectly wrote it as "vile" tester and hastily corrected it to "evil" tester. And now I have to live with this vile tag forever. Poor poor pitiful me.
3. I got drunk and, well... you know how it goes.
4. <insert your plausible answer here>

EVIL

Q3. DEAR EVIL TESTER...

In the past I have worked with project managers who have only pretended to manage. Just in case I come across this sort again in the future, I want to know; Should I only pretend to test too as a form of self-preservation? - Eliza

Ah, a philosophy question. How do you know you don't already 'pretend' to test? What does it mean to 'really' test something? But since I don't do philosophy. I can answer simply. "NO".

If you short change yourself then that isn't self-preservation. It is allowing your skills and integrity to slowly rot, wither and die. It is condemning yourself to victim hood as a response to other people's actions. Don't do that to yourself.

Always. Always. Work to the highest level that you can be proud of. That is an act of self-preservation. That is a process of self-maintenance. That is a coping strategy which allows you to weather the storms of incredulity that attempt to swamp you on project after project.

I try very hard to get in the habit of evaluating myself. Not in terms of the actions of others, or in terms of their expectations of me, or in terms of a 'generic' tester. I try to evaluate myself in terms of my expectations of me. And I try to continually raise my expectations.

I'm not going to tell you to stop evaluating people and building perceptions of them, because that is an essential skill to office survival. We have to learn who the predators are so that we either keep our distance from them, or hunt them in a pack.

UNCLE EVIL

Q4. DEAR EVIL TESTER...

Why is database testing so painful? - Mike

Are you confusing database testing with physical torture? If so, then you'll want to read my "Evil Tester's Handy Tips Summary Guide to Telling The Difference Between Physical Torture and Database Testing" below. In database testing we tend to do some, or all, of the following:

- insert data into a set of tables
- delete records when the system isn't expecting it
- generate data randomly using an automated tool
- stress the database with 1000s of transactions
- use powerful ETL tools on the database

Please don't confuse the above with actions on your own person, so for a pain free testing experience, never do any, or all, of the following:

- insert anything into yourself at work
- ask people to steal your chair when you try to sit down
- setup a machine to randomly shoot cutlery at you
- ask your team to help you run the gauntlet
- use power tools on yourself

Don't worry Mike, we've all been there. I've still got the scars to prove it. But by following my handy tips above I'm sure you'll start to find database testing less painful.

AUNTIE EVIL ☐



IF THE CUSTOMER NEVER SEES A BUG, DOES IT EVER EXIST?



JESPER LINDHOLT OTTOSEN - 10 JAN 2012

Same problem: Is the cat dead? <http://testing.gobanana.co.uk/?p=707> by Adam Brown

JAMES O'SULLIVAN - 10 JAN 2012 (EDITED)

Very metaphysical. I'd say that a bug doesn't exist unless it's observed by someone :)

TIM HALL - 10 JAN 2012

I've seen code that had been in production for two years, and could never possibly have worked. Since the customer never reported the bug, concluded that they never actually used that feature.

Of course, it may be that an end-user has problems with the feature, but was too low in the organisation's food chain to report it.

MARK TOMLINSON - 11 JAN 2012

Bug? What bug? :-)

NOLAN MACAFEE - 11 JAN 2012

Sort of like "If a tree falls in a forest and no one is around to hear it, does it make a sound?"

ADAM BROWN - 11 JAN 2012

Thanks for the link, Jesper :)

I would say not. If it has no impact on life and limb, user experience, systems, software etc and the only person who cares about it is the tester, I would say it's not a bug.

JESPER LINDHOLT OTTOSEN - 11 JAN 2012

Indeed Adam. Also keeping in mind that a bug is something that bugs - someone who matters :-)

ADAM BROWN - 11 JAN 2012

^ I think I'll be using that as a quote sometime.

ADAM KNIGHT - 11 JAN 2012

A light hearted point but raises an interesting issue over the oft quoted 'bugs found in test to bugs found live' ratio: if those bugs found in test would not have been encountered in live use, have you

focussed on the wrong areas?

TIM HALL - 11 JAN 2012

+Adam Knight I've often encountered that after I've raised a bug - Frequently get asked whether a real user would ever attempt to do that. But real users will often use (or try to use) a system in ways the original designers didn't intend rather than request a new feature.

ADAM KNIGHT - 11 JAN 2012

+Tim Hall good point, this is where I think great testers differentiate themselves, not just identifying bugs but also putting forward a compelling case as to whether they should be fixed as a priority. My point was primarily to highlight that bugs are complicated and that apparently simple metrics based on them can be misleading.

SEAN MORLEY - 11 JAN 2012

I would say yes, it does exist - with the proviso that it can actually be hit. Of course it can depend on how you define terms. You might define a bug as a found fault; not found = no bug. But one way to define quality is the likelihood that a user will hit a bug/fault. Thus a lurking bug is a bug nonetheless.

Great points above about compelling cases and testing the right stuff.

Reminds me of my days in hardware test generation. We modeled physical faults, and could grade a test program for coverage. The circuit design might be such that a fault could not be found. We could algorithmically prove that, and adjust the grade. And then there was the argument about faults which would never be hit based on intended usage. Interesting stuff.

ROSIE SHERRY - 11 JAN 2012

Some great comments :)

+Nolan MacAfee yup, 'the tree falling...' is where the idea originally came from.

One question though - how do you know if a customer has found a bug? If they report it, then of course you know, but am sure many don't bother reporting bugs. I know I don't. I occasionally blog about them for fun though :)

NICOLA SEDGWICK - 13 JAN 2012

How do you define customer? The people with control of the chequebook, the people who define the system and the people using the software at the coalface are often discrete sets with little to no overlap and each set tends to have a very different idea of what constitutes a bug.

ROSIE SHERRY - 13 JAN 2012

+Nicola Sedgwick yeah, guess it depends on the scenario. Types of customers vary between types of projects.

But if none of them find a bug, does it exist? :)

ROSIE SHERRY - 13 JAN 2012

I'd love to publish some of these answers in the next TTP - really enjoying the thread...keep some responses coming!

DAVID RAMSAY - 13 JAN 2012

Under the premise that if a bug isn't seen then there isn't a bug, surely that means if we do not test there are no bugs and the software can be shipped ;-)

RASMUS KOORITS - 13 JAN 2012

The question is difficult to answer because the answer will depend on the current state of the project the bug resides in:

1) Let's say it's early in development. We found a bug. We fixed it! Thus, the customer never sees it. It used to exist, but not anymore. Is that a valid answer?

2) Let's say we find a bug that the customer will never see (it's somewhere in the back-end or we create quick and dirty fixes). Now, even though the customer never sees it, we still have to take its existence into account - because some change to the code base might make the bug appear again.

3) Same thing with dead areas of code. Imagine a bug exists in lines of code that are actually commented out. Now, we still have to know that these lines are trouble; if someone chooses to reanimate this code some day.

4) Can a customer NOT see a bug yet still be troubled (bugged) by it? Imagine a performance issue. The customer (or even everyone in the development project!!) might think that since the tool is huge and does lots of work, it's supposed to be slow. When in fact it could be really fast if not for this bug in performance nobody knows about?

Also, a question about your terminology: is there a difference for you between customer and user? Imagine software a bank accountant uses to deal with a customer. The user might have lots of

Continued on page 26



Continued from page 25

bugs, but if he finds the correct workarounds, the customer experience can be flawless.

ANDERS DINSEN - 13 JAN 2012 (EDITED)

The answer to the question depends on our use of daily language. +Jesper Lindholt Ottosen points out the similarity of the question to the problems in (quantum) physics, +Nolan MacAfee compares with the philosophical question about the falling tree making a sound or not if noone observes it.

If we define bugs as “something that bugs someone”, bugs don’t exist until they are observed,

since unobserved bugs don’t bug anyone. If that’s how we see it, the answer to the question is NO. But thinking a bit further along this line, if that’s how we define bugs, testing /produces/ bugs.

But I usually hear testers talk about testing as an activity which /discovers/ or /finds/ bugs - and the testing activity is carried out in order to find as many as possible. In that case, the answer to the question is YES: The bug is there even if nobody finds it.

So there’s no clear answer to this question - it depends. On what? It depends on how we talk about bugs. By observing how we use language, we can learn a lot about the way we think about the work

we do, and thereby about the way we work.

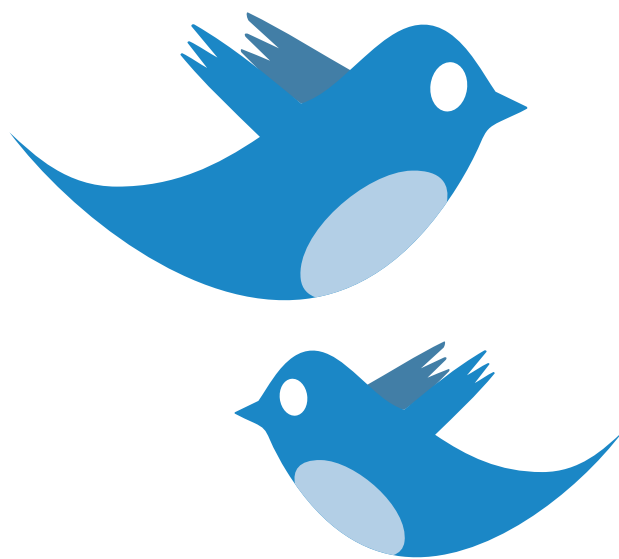
+Rasmus Koorits regarding your item 1: Is a fallen tree a tree? is a fixed bug still a bug? :)

JEFF LUCAS - 13 JAN 2012

This is Schrödinger’s issue - it is both a bug and an enhancement until it is observed. I think we have discovered the unified theory of intermittent defects.

ROSIE SHERRY - 25 JAN 2012

Another mention of this for reference - <http://jabbawocky.posterous.com/if-users-do-not-encounter-any-bugs-do-they-ex> □



FROM THE TWITTERSPHERE

@TESTINGCLUB

“if you find a great many bugs, does that mean the #testing is effective or the product is lousy?” - GMWeinberg #QuestionsCountingCantAnswer

@MIGHTYSQUID

@QualityFrog @testingclub Oh, I always plan to plan. :)

@QUALITYFROG

@MightySquid @testingclub a test plan is not the same as a test plan document. the planning is everything, the document is nothing. :)

@MIGHTYSQUID

@QualityFrog @testingclub In my experience no one actually reads my Test Plans but I see your point. That’s why open comm is important.

@QUALITYFROG

@MightySquid @testingclub in my experience, explicit pre-defined entrance criteria for testing does more harm than good. #TestingIsNotAPhase

@MIGHTYSQUID

@QualityFrog @testingclub Agreed. There can be a metric such as # of failed cases per time period but common sense should prevail. □



THE TESTING PLANET

TIME FOR A CHANGE

The Testing Planet is two years old this issue and, although we didn’t go all out in our anniversary celebrations (we’ll do that when we’re five!), we are making some changes. Since you’re reading this, there’s a good chance that you’re already aware of what those changes are and why we feel they’re necessary, but we want to make sure the record is straight and give you, our loyal readers the opportunity to respond to and feedback into the journey.

Supreme Testing Club Overlord Rosie Sherry has already blogged ¹ about some of the challenges faced by the editorial team. In summary:

- The review and editing process is time consuming and demands substantial effort from everybody concerned – the editorial/design team and the article authors.
- Doing the job properly and producing a publication that the testing community can be proud of costs a lot of money.
- The money required to continue publishing the Testing Planet can only be generated by:
 1. Selling much more advertising space, and
 2. Charging a fee for the printed and/or digital publication.

We’re left with a choice then; do we go for option [1] and abandon the unbiased editorial principles by which we currently try to operate? Or do we strive for option [2] and hopefully encourage our readership to support The Testing Planet cause and invest in a more sustainable, community led, professional testing newspaper?

We think option [1] equals selling out. If we choose this option, The Testing Planet is likely to be filled with vendor tool based articles and white papers explaining why Company X’s proprietary testing methodology will lead to 0 defects and 100% satisfied customers in the next

release. *Yes, we do actually get papers like this.*

At this point, we have to ask ourselves – “do we understand our readers?” As a relatively new addition to The Testing Planet team ², I think we do. I believe that our readers want us to be something different. I’d assert that they’re hoping for a daring and vibrant perspective on the culture, practices, and skills of software testers and the organisations, products and paradigms within which they work. I’d like for us to be able to achieve that lofty goal, but we need your help to make it happen.

We need your continued support and input. Our mission is to serve our readers by providing them with a publication within which their passion for the software testing craft can be explored and encouraged.

We want to continue to inspire, inform and educate you, intrepid tester. Will you come with us on our journey? Will you take this opportunity to let us know what you think? Will you tell us what you would like to see in future editions of The Testing Planet? Would you like to get involved? Contribute an article? Write a column? Share a thought, an opinion or write a letter to the editor?

You can let us know your thoughts via feedback@softwaretestingclub.com or via @TestingClub using the #TestingPlanet Twitter hashtag.

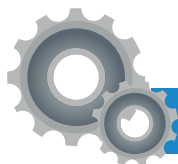
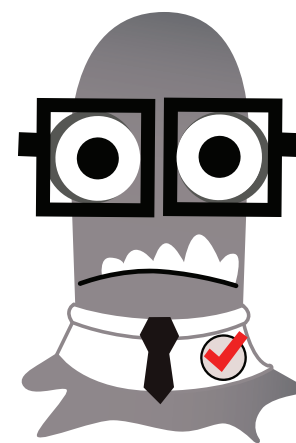
Thanks for coming along on the journey so far. I hope to see you farther down the line. And if you do like what we’re achieving here, please spread the word! □

REFERENCES

1. <http://blog.softwaretestingclub.com/2012/02/time-for-a-change-2-years-of-the-testingplanet/>
2. <http://www.thetestingplanet.com/2011/12/introducing-our-new-editor/>



TESTING TIPS



ROB LAMBERT

IE9 comes with its own in built backwards compatibility mode for older browser versions. This means you can open IE9 and test against old (7 & 8) versions of IE.

How about getting a dead-link checking tool (like Xenu) built in to your Continuous Integration or Overnight Build Process? That way you can check for the presence of dead-links after an automated test run.

In several of the modern tabbed browsers it is possible to open up multiple tabs and then drag a tab out of the main browser “window” to create two ‘Windows’ operating under one browser session. This makes it easier to see both tabs on the same view.

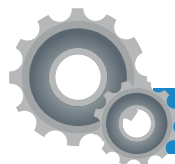
All browsers now come with basic Developer Tool-kits. Try browsing the options sections in your favourite browser and opening the Developer Tools section. You’ll see plenty of interesting developer features that will help you in your quest as a Tester. Why not also try supplementing them with monitoring plugins?

If your products are on the Salesforce App exchange you get a licence for the Burpsuite security tool for free!

It’s worth spending an hour each week seeking out cool browser extensions. Some of these extensions can extend your capabilities so dramatically that they become invaluable to your day to day testing.

Most browsers “back” functionality will allow you to see numerous historical pages. Try skipping back three or four pages rather than just to the last page. Can you gain access and see things you shouldn’t? Can you change data and resend that page? What happens to the underlying data?

There are extensions for all sorts of activities you might want to do whilst browsing. Well worth checking out GreaseMonkey scripting too as this allows you to script out behaviour in the browser. Very powerful stuff.



SIMON KNIGHT

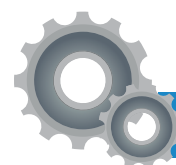
Identify “Steel Thread” scenarios that enable automated testing of large chunks of functionality in a short space of time. Often when creating automated tests you’ll be focused on a specific area of functionality, e.g. a set of fields. Creating many such tests and their respective variations can be time consuming and often leaves gaps in coverage due to time constraints when trying to complete all of your tests. Identifying a few higher level, end-to-end processes (Steel Threads name and concept coined by Lisa Crispin - “Experiences in Test Automation”) means that more coverage is achieved within the timeframe available.

Always strive to insert your test data directly into the database rather than using a test tool. You never know when the underlying database schema’s will change, leaving you without data and potentially without a data creation mechanism if your automation suite requires maintenance. Create your data via inserts in the first instance, and save yourself a headache further down the line!

Use JMeter for “smoke” tests. In addition to being a decent load & performance testing tool, JMeter can also be used for functional testing. Use JMeter HTTP samplers with Response Assertions across your website to validate basic functionality before carrying out further exploratory testing.

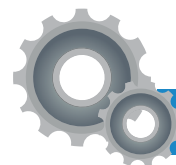
Identify security flaws using capture, modify and replay tools. Did you know that you can circumvent field validations on your web application? Here’s a simple way around them: use a capture and modify tool to manipulate browser requests before they

reach the server. There’s lots of them out there; e.g. the Burp Suite application, Firefox plugin Tamper, and the recent OWASP addition ZAP!



STEVELAND DANIELS

“For people that have access to a Windows 7 machine, there is a utility called Problem Step Recorder. It is a utility that records your steps and outputs them to a nice HTML file (*.mht). It can record the exact keystrokes and clicks that you entered or performed. It automatically captures the screenshots and other technical information. To access it, navigate to the run command and type psr. The Problem Step Recorder will then load.”

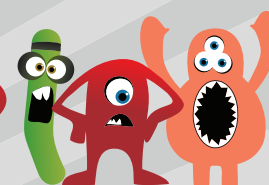


LEON VAN STADEN

The maturity level of both the AUT (Application Under Test) and the Tester plays a critical role when testing software. If and when there is no documentation then spend as much time as possible to understand the System or Application you are dealing with, leaving no stone unturned. While you carry out your investigations be sure and do those tedious yet all important screen shots (before and after) and ask questions whenever you hit an ambiguous function. Simply follow a bottom up approach. If testers tell me it would be impossible to test with proper documentation, I then go ahead and do the testing on their behalf. These are the people who look for a job but pray not to find any! Hardcore testers are a rare species. □

WWW.SOFTWARETESTINGCLUB.COM

Software Testing Club
a community for software testers



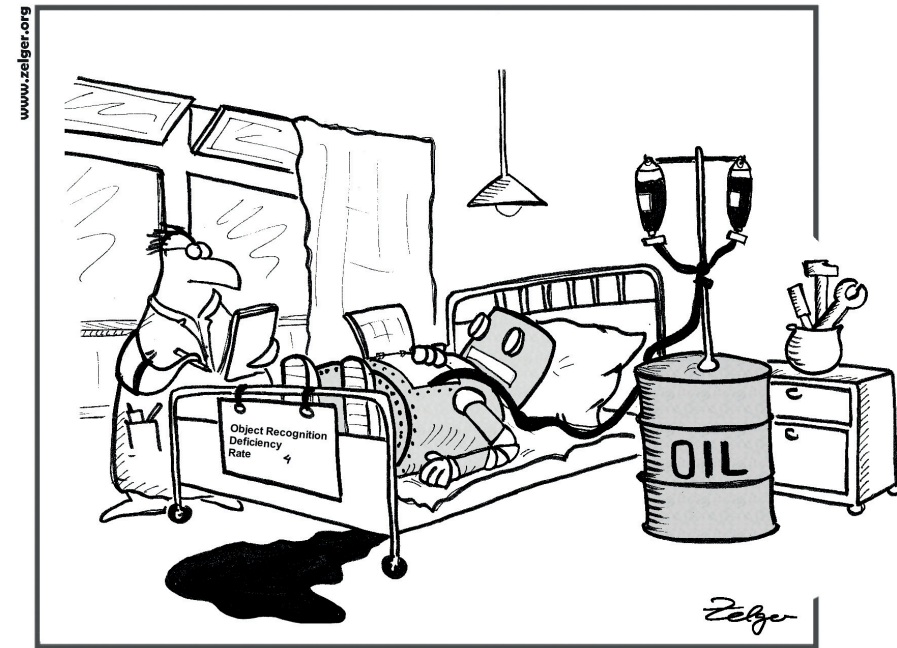
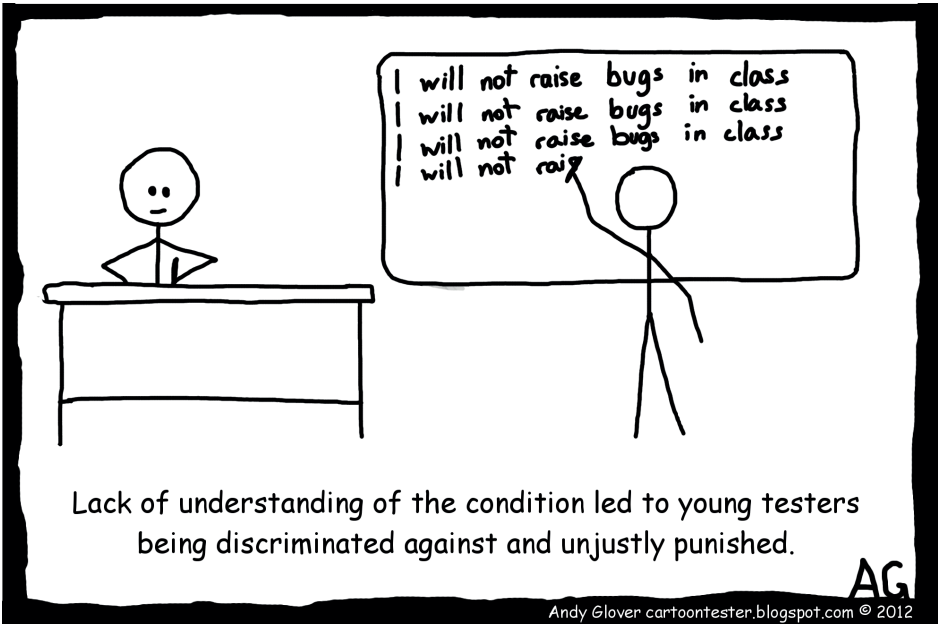
Micromanager is Not Transparency! #pmot icont.ac/UCVP @johannarohtman

Your Guide to the Software Testing Club




CARTOON CORNER BY ANDY GLOVER A.K.A THE CARTOON TESTER - <http://cartoontester.blogspot.com/>

the cartoon corner



"Here is the diagnosis: You suffer of an acute UI object recognition insufficiency...We will give you an infusion and in two days you can run and traverse along your recorded test paths again".





CO-CREATING
SMARTER TESTERS

EDUCATION • COLLABORATION
EVENTS

WWW.MINISTRYOFTESTING.COM

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory

TEST TOOLS & SOFTWARE



GEMINI

GEMINI

Gemini brings versatile test management, bug and issue tracking to your team. Sign up to our cloud-based offering or install locally. Join the new generation in software project management with Gemini – no hidden extras or crazy pricing. 3 Users FREE – No Gimmicks – Full Edition. www.geminiplatform.com

TestRail

TESTRAIL

TestRail – Test Case Management Software for QA and Development Teams. Comprehensive web-based test case management software to efficiently manage, track and organize your software testing efforts. www.gurock.com/testrail

PractiTest

PRACTITEST

Practitest is a SaaS-based Test Management Solution that supports the entire QA lifecycle, including requirement & issue tracking. www.practitest.com

ReQtest

REQTEST

ReQtest is an easy to use bug tracking software, available in the cloud 24/7. It empowers teams to work more efficiently and gives decision makers meaningful data on progress made. ReQtest includes a requirement management module which is tightly integrated with the bug tracking features. www.reqtest.com

kalistick

INNOVATION FOR AGILE QA

KALISTICK

Kalistick gives testers a new solution to design efficient test strategies focusing on business risks. Our unique technology analyzes test cases footprints and functional changes to select the most relevant test cases. Discover how to move one step ahead in testing efficiency. www.kalistick.com



BugDigger

BUG DIGGER

BugDigger removes the hard work from web site bug reporting. With the help of a browser add-on, automatically captures and uploads: – web page screendump optionally annotated using built-in editor, – environment details, and – web site usage history. Even busy or inexperienced testers can create useful bug reports instantly. BugDigger integrates with JIRA, Basecamp, Pivotal Tracker, FogBugz, Unfuddle, Redmine and others. www.bugdigger.com



TEST WAVE

TESTWAVE

TestWave is a next generation test management tool implemented as Software as a Service (SaaS). It can be deployed instantly and you only pay for what you use. TestWave is designed for both Test Managers and Testers, and provides requirements, test planning, test execution and defect tracking. Intuitive graphs report testing data in real time. Reduce your costs and unleash the power of SaaS with the cloud's first fully extensible test management tool. Learn more and sign up for a free 30 day evaluation: www.testwave.co.uk

PARASOFT. SOAtest™

PARASOFT SOATEST

Parasoft SOAtest automates web application testing, message/protocol testing, cloud testing and security testing. Parasoft SOAtest and Parasoft Load Test (packaged together) ensure secure, reliable, compliant business processes and seamlessly integrate with Parasoft language products (e.g., Parasoft Jtest) to

help teams prevent and detect application-layer defects from the start of the SDLC. Moreover, Parasoft SOAtest integrates with Parasoft Virtualize to provide comprehensive access to traditionally difficult or expensive to access development and test environments. Parasoft SOAtest provides an integrated solution for: End-to-end testing, Environment management, Quality governance, Process visibility and control. www.parasoft.com

TESTPLANT

TestPlant develops eggPlant the leading user interface test tool that creates an abstraction of a GUI for any device type, enabling automation of screen-based testing through 'search and compare'. Download now. www.testplant.com

XSTUDIO

XStudio is a free ALM/test management solution allowing to manage requirements/specifications, scrum projects, Automated/manual tests, campaigns and defects. An LGPL SDK is also included to interface with proprietary tests. www.xqual.com

TESTLODGE

TestLodge is an online test case management tool that allows you to manage your test plans, requirements, test cases and test runs with ease along with issue tracker integration. www.testlodge.com

TESTOPTIMAL

TestOptimal – Model-based data-driven test design and test automation to improve test coverage, enable rapid response to changes and reduce test maintenance cost. www.testoptimal.com

LOADSTORM

LoadStorm – The lowest cost and easiest cloud load testing tool. Free account for 25 users. Test up to 100k vusers. Real-time graphs with key performance metrics. www.loadstorm.com

SOFTWARE TESTING TRAINING

SKILLS MATTER

SKILLS MATTER

Skills Matter supports a community of 35,000 Software Professionals with the learning and sharing of skills to write better software. Find hundreds of meetups, talks, conferences, skillscasts and workshops on our website: www.skillsmatter.com

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory



New to software testing? Read this - <http://bit.ly/zFdtW>

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory

SOFTWARE TESTING SOLUTIONS

ElectroMind

YOUR SOFTWARE QUALITY MATTERS

ELECTROMIND

ElectroMind offers training, consulting, coaching and mentoring services to the software testing community. Through strong relationships with world-class testing experts, built up over several years, ElectroMind delivers niche training products, test process improvement consultancy and innovative people skills development programmes. Our consultants are comfortable using both traditional and Agile testing methodologies with experience in several industry sectors including financial services, telecommunications, online retail, travel, mobile and digital media. Through strategic partners, ElectroMind can offer performance engineering services including load and stress testing. Our overall philosophy is simple. We believe your software quality matters. www.electromind.com



TEST HATS

Test Hats are an independent software testing services provider, with offices in the UK and Spain. We provide a full range of testing services including System, Performance and Security testing along with specialised Consultancy and Training. For near-shore testing our Test Lab is fully equipped with a range of desktop and mobile platforms, testing software and tools, allowing us to provide a quality service at a competitive price. Visit our website to learn more about Test Hats and our services. Get in touch today to talk about how we can help test your projects. www.testhats.com



THE TEST PEOPLE

The Test People delivers the best, most innovative, highly technical and competitive performance engineering and test service available today. Based upon our extensive experience, TTP can

deliver tailored services to address all aspects of the functional and non-functional test lifecycle, including highly specialised performance engineering and test automation services including automated build and continuous integration solutions. TTP are at the forefront of utilising the cloud for test and load environments, with significant experience in open source and the major commercial toolsets whilst also coming armed with our own performance and automation frameworks. www.thetestpeople.com

ORIGINAL SOFTWARE

Original Software - With a world class record of innovation, Original Software offers a solution focused completely on the goal of effective quality management. By embracing the full spectrum of Application Quality Management across a wide range of applications and environments, products include a quality management platform, dynamic manual testing, robust test automation and test data management. More than 400 organisations operating in over 30 countries use Original Software solutions. Amongst its customers are Coca-Cola, Unilever, Barclays Bank, HSBC, FedEx, Pfizer, DHL and many others. Visit www.origsoft.com/solutions for more information.



brainual software testing services

MOOLYA

Moolya is a new generation software testing services company headquartered in Bangalore, India founded in 2010. Our focus is to help business move forward. We believe in helping our clients to make great products that wow their customers. That's when we win. We are context driven testers highly skilled at exploratory testing, SBTM, small "a" agile testing and check automation. How can we help you win smiles on your customer's face? sales@moolya.com / www.moolya.com



REVOLUTION IT

Revolution IT is the leading Quality Assurance

and Testing, management consulting firm in Asia Pacific. We help our clients deliver IT projects and have core offerings across Project Management, Requirements Management and Application Testing. We have over 250 staff and offices in Melbourne, Sydney, Brisbane, Canberra, Adelaide and Singapore. Our offering includes delivery consulting, methodologies, tool solutions and training. We have strategic partnerships with HP software, IBM Rational, Oracle, Agile Academy and SAP. With HP we have been the leading HP Software Platinum Partner for 4 years running and the leading reseller, 1st line technical support, training and services partner. www.revolutionit.com.au

INDEPENDENT TESTERS, TRAINERS AND CONSULTANTS



ANNE-MARIE CHARRETT

Anne-Marie Charrett is a testing coach and trainer with a passion for helping testers discover their testing strengths and become the testers they aspire to be. She offers a blend of online coaching and training on Exploratory Testing, Career Management and motivating testers. Anne-Marie is currently working on a coaching testers book with James Bach, due out late next year. www.testingtimes.com.au

COMMUNITIES, CONFERENCES AND NEWS

EuroSTAR

EUROSTAR

EuroSTAR is Europe's premier software testing event and will be taking place this year in Manchester, UK from November 21 – November 24. At EuroSTAR 2011, the leading names in testing will meet for an intensive 3-4 days of learning, networking, discussion...and a few extracurricular activities! Attendees can choose from numerous thought-provoking presentations, intensive tutorials, interactive sessions and inspirational keynotes. Plus, visit Europe's largest software testing exhibition which will be showcasing the leading companies in the industry. The EuroSTAR Team hopes to see you in Manchester later this year for what will be a fantastic, fun and innovative conference! www.eurostarconferences.com □

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory



Why was there a bug in the computer? It was looking for a byte to eat. - @steveo1967



MINISTRY OF TESTING
CO-CREATING SMARTER TESTERS

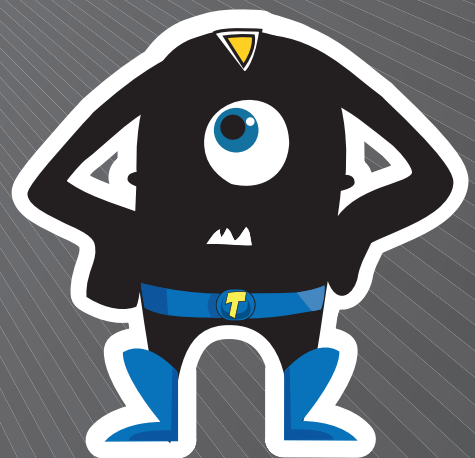
EDUCATION



COLLABORATION



EVENTS



WWW.MINISTRYOFTESTING.COM