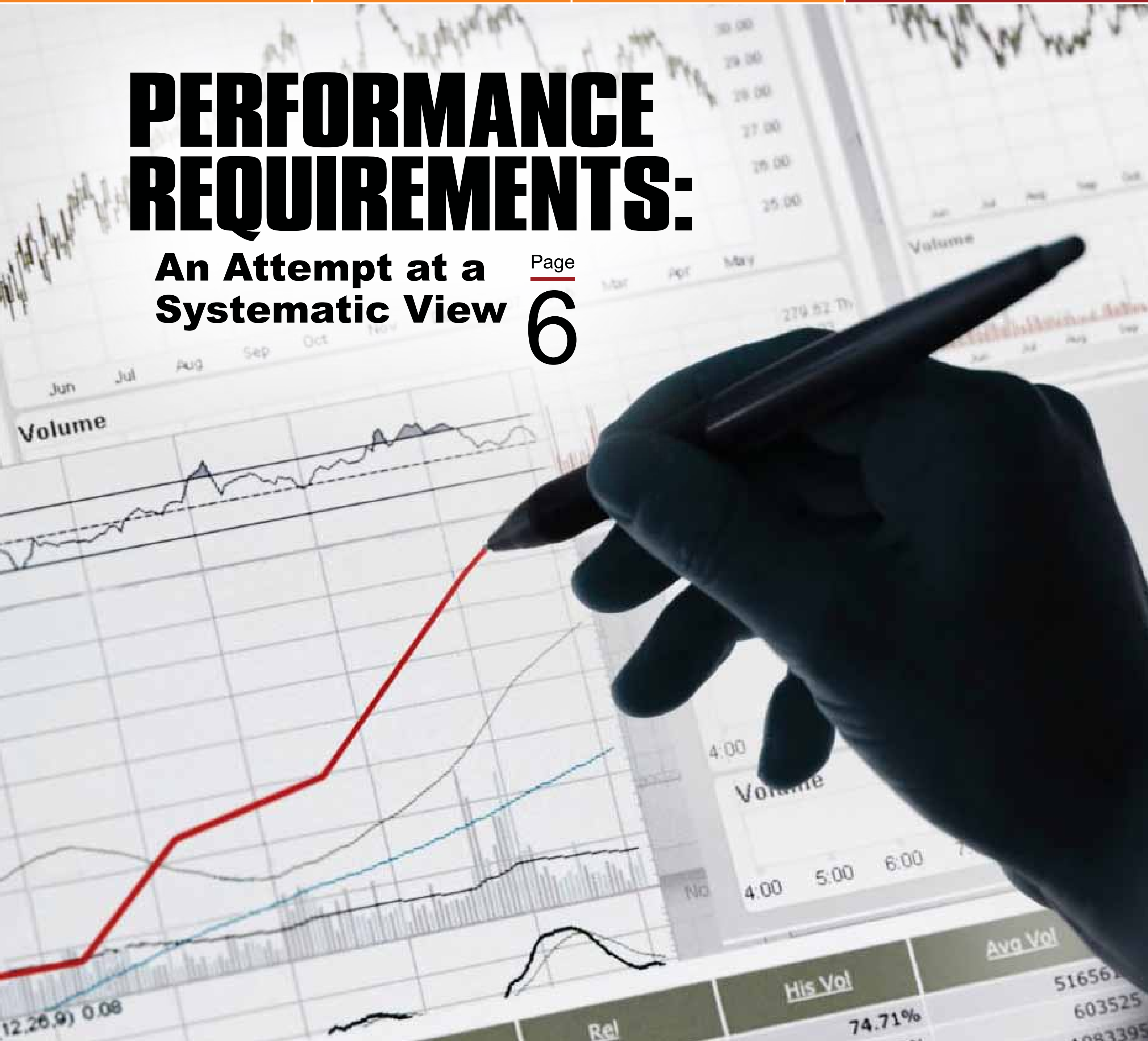
Software Test & Quality Assurance

ST&QA MAGAZINE

STP SOFTWARE TEST PROFESSIONALS

VOLUME 9
ISSUE 1
DEC/JAN
2011–2012

www.softwaretestpro.com



PERFORMANCE REQUIREMENTS:

An Attempt at a Systematic View

Page 6

STP SOFTWARE TEST PROFESSIONALS

877.257.9531

Live Interactive Professional Development Without Ever Leaving Your Home or Office

STP Online Summits are live topic-specific events delivered entirely online for three and a half hours per day over three consecutive days. Speakers present their sessions live in real time, right on your computer screen. You can listen to and watch their presentations, as well as ask questions and get instant answers.

Why Attend?

- Summit programs are created for testers, by testers.
- No travel required.
- Network and interact live with speakers and participants.
- Sessions are spread out over three days in order to minimize the impact on your daily schedule.

Date	Time	Topic
2/21/12 – 2/23/12	10:00AM – 1:30PM PT	**Agile Transitions:** Tips to Seamlessly Integrating Testing and Testers to Achieve Business Goals
4/10/12 – 4/12/12	10:00AM – 1:30PM PT	**Test Management:** Bridging the Gap Between the Tests and the Stakeholders
6/5/12 – 6/7/12	10:00AM – 1:30PM PT	**Mobile Devices and Applications:** What's the Same, What's Different, and What That Means for Testing

$245 per event or $195 per event if you register now!

For more information email info@softwaretestpro.com or call 877.257.9531

www.softwaretestpro.com/onlinesummit

Because your web site should be in better shape than you are.

Training the NYC Marathon live website to be the fastest in the world for three years running.

New York Road Runners is the organization behind the world-famous New York City Marathon, which sends 40,000 runners across the city's five boroughs for 26.2 miles each November. Fans register to follow their favorite runners online, and these individual status pages are updated continuously until the exhausted and exhilarated runners cross the finish line, 20-30 participants per second.

For the past three years Web Performance has load tested the servers that provide live race information. We test and tune the capacity of the Athlete Tracking Website up to a level of 100,000 concurrent users, and the site handles the live race traffic without breaking a sweat.

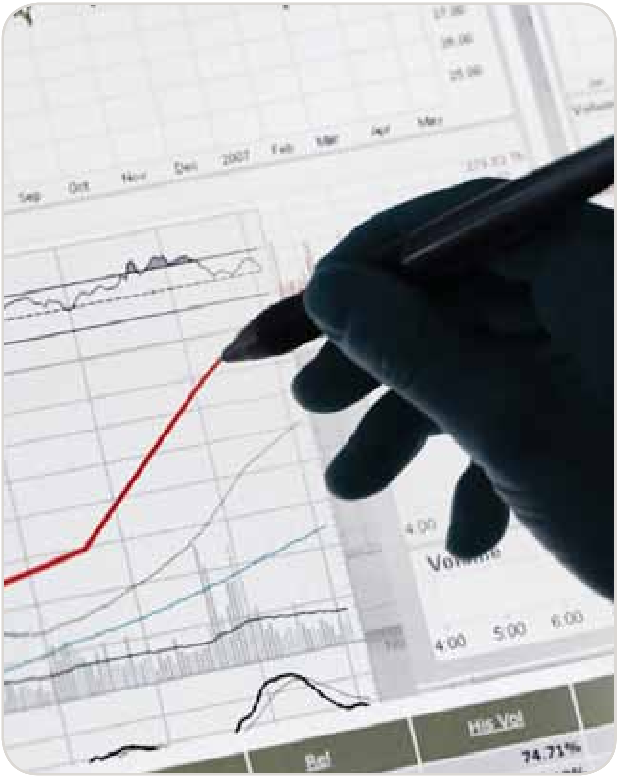
Load Testing Software & Services

webperformance.COM

CONTENTS

Volume 9 / Issue 1

@SoftwareTestPro SoftwareTestProfessionals



Editor-in Chief: Rich Hand
rhand@softwaretestpro.com

Creative Director: Scott Hanson
shanson@softwaretestpro.com

How to reach ST&QA
Letters to the editor may be sent to:
editor@softwaretestpro.com

About Software Test Professionals (STP)
STP provides more than 50,000 software professionals with information, education, training and professional networking opportunities.

724 S Tejon Suite C • Colorado Springs
Colorado, 80903 • 719.476.0301

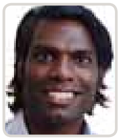
About ST&QA
ST&QA is published bi-monthly by STP (Jan, Mar, May, July, Sept, Nov). It is a digital magazine geared towards the software test and quality assurance professional. Subscription to ST&QA is an STP Pro Membership benefit. If you would like to subscribe to ST&QA, visit softwaretestpro.com and join the Pro Membership.

Back Issues of ST&P and ST&QA
Pro Members may access electronic back issues at www.stqamag.com.

Editorial Submissions
Visit softwaretestpro.com to learn more about submitting editorial to ST&QA.

ST&QA Magazine (ISSN #1548-3460) (USPS #23771) is published bi-monthly by Redwood Collaborative Media (724 S. Tejon Street, Suite C, Colorado Springs, CO 80903). Periodicals postage paid at Colorado Springs and additional mailing offices. The price of a one year subscription is US $69 for postal addresses within North America; $119 elsewhere. ST&QA Magazine is a registered trademark of Redwood Collaborative Media. All contents copyrighted 2010 Redwood Collaborative Media. All rights reserved. POSTMASTER: Send changes of address to Software Test Professionals, 724 S, Tejon Street, Suite C, Colorado Springs, CO 80903. To contact ST&QA Magazine subscriber services, please send an e-mail to info@softwaretestpro.com or call 719.476.0301.

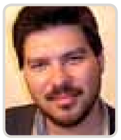
Contributors

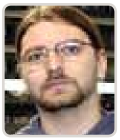

HenrikANDERSON
Founder – House of Test

BernieBERGER
VP Financial Services

DevyaniBORADE
Software Tester – web solutions provider near London

BudhadityaDAS
Test Consultant, Performance and Automation – Mindtree Limited


AlbertGAREEV
Dedicated Testing and Test Automation Professional


FelipeKNORR KUHN
Software Engineer – São Paulo, Brazil

MichaelLARSEN
Lone Tester – SideReel.com

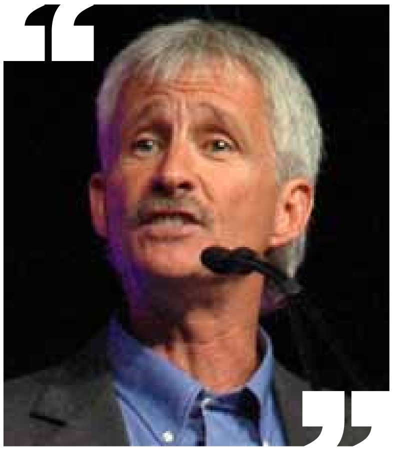
BrettLEONARD
Software Testing Group Manager – Construction Software Technologies

Brian J. NOGGLE
Freelance Software Testing Consultant – Jeracor Group LLC.

AlexPODELKO
Consulting Member of Technical Staff – Oracle

The economy will be a challenge...

Rich HAND



"By utilizing our Conference Program Board, we hope to become your trusted source for industry information." Rich HAND

As we face the New Year there are going to be great challenges and opportunities in the software industry. More devices, more distribution, and more adoption of technology will continue to stress our industry. Pressure to reach the market faster will stretch QA to the limits and unfortunately will break it on occasion. Although there has been a lot of discussion around the role of testing being dead, I would not predict that will happen in 2012.

The economy will again be a challenge but also a huge opportunity for our industry. The reality is that even though consumers are adopting new mobile technologies at breakneck speed, many organizations including the Government have huge opportunities to create efficiencies by upgrading and improving both software and infrastructure as cost pressures drive this need with greater intensity. This means our industry sits at the cutting edge of efficiency and growth in what continues to be forecast as a difficult economic climate in 2012.

In order to support the industry's efforts in 2012, we here at STP will continue to bring user based content to the Software Test community in the ST&QA Magazine, the Insider, and the Test & Quality Assurance (TQA) publications. STP's signature will continue to be the two conferences we produce each year that we believe are a testament to quality. We serve a very critical audience and we have heard our attention to detail soars above the other events in our industry. We will be hosting the Spring conference in New Orleans, March 26th through the 29th.

By utilizing our Conference Program Board made up of practitioners to help develop conference programs, publishing authors that want to engage in the debates and share their knowledge, and creating online summits focused on the hottest trends in the industry, we hope to become your trusted source for industry information.

We have kicked off 2012 with articles from **Alex Podelko**, *Performance Requirements: An Attempt at a Systematic View*, *Leveraging Web Analytics for Effective Web Applications Testing* by **Budhaditya Das** and **Paul Fratellone**, our regular feature *Ask the Tester* by **Matt Heusser**, interviewing **Adam Goucher**, and *Managing Test Data* by **Devyani Borade**. The great articles continue with *Scrum and SBTM, Let's Get Married!* by **Henrik Anderson**. **Bernie Berger** is back with his *Day in the Life of a Software Tester* (part 3), *Guerrilla Bug Reporting* by **Brett Leonard**, *Coming to TERMS With Test Automation* by **Michael Larsen** and **Albert Gareev**, *Trouble Tickets Are Your Business* by **Brian Noggle**, *Running Your Own Selenium Hub* by our friend from Brazil **Felipe Knorr Kuhn**, and our favorite cartoonist **Torsten Zelger**.

We are jam packed!

If you have an interest or expertise you would like to contribute to the community I am encouraging you to do so. We are embarking on another year of our profession and maybe it's time for you to get more involved in shaping the conversation. It is our goal to get many voices in the conversation about software testing. Our publications are a great way to get noticed.

I wish everyone a successful 2012 as we work to help you accomplish your goals. Thank you for your past support and continued participation in the New Year!

[illegible]

Rich HAND
Director of Membership & Publications

Part II

PERFORMANCE REQUIREMENTS: An Attempt at a Systematic View

In the May/June 2011 issue of ST&QA Magazine, in the first part of the article, we discussed the most important performance metrics. Now we will discuss all stages of the performance requirements process, which include elicitation, analysis, specification, and validation, according to the IEEE Software Engineering Book of Knowledge (SWEBOK).

by AlexPODELKO

IEEE Software Engineering Book of Knowledge defines four stages or requirements[1]:

1. **Elicitation:** Identifying sources and collecting requirements.
2. **Analysis:** Classifying, elaborating, and negotiating requirements.
3. **Specification:** Producing a document. While documenting requirements is important, the way to do this depends on software development methodology used, corporate standards, and other factors.
4. **Validation:** Making sure that requirements are correct.

Let's consider each stage and its connection with other software life cycle processes.

Elicitation

If we look at the performance requirements from another point of view, we can classify them into business, usability, and technological requirements.

Business requirements come directly from the business and may be captured very early in the project lifecycle, before design starts. For example, a customer representative should enter 20 requests per hour and the system should support up to 1000 customer representatives. Translated into more technical terms, the requests should be processed in five minutes on average, throughput would be up to 20,000 requests per hour, and there could be up to 1,000 parallel user sessions.

The main trap here is to immediately link business requirements to a specific design, technology, or usability requirements, thus limiting the number of available design choices. If we consider a web system, for example, it is probably possible to squeeze all the information into a single page or have a sequence of two dozen screens. All information can be saved at once in the end or each page of these two-dozen can be saved separately. We have the same business requirements, but response times per page and the number of pages per hour would be different.

While the final requirements should be quantitative and measurable, it is not an absolute requirement for initial requirements. Scott Barber, for example, advocates that we need to gather qualitative requirements first[2]. While business people know what the system should do and may provide some numeric information, they are usually not trained in requirement elicitation and system design. If asked to provide quantitative and measurable requirements, they may finally provide them based on whatever assumptions they have about the system's design and human-computer interaction, but quite often it results in wrong assumptions being documented as business requirements. We need to document real business requirements in the form they are available, and only then elaborate them into quantitative and measurable efforts.

One often missed issue, as Scott Barber notes, is goals versus requirements[2]. Most of response time "requirements" (and sometimes other kinds of performance requirements,) are goals, not requirements. They are something that we want to achieve, but missing them won't necessarily prevent deploying the system.

In many cases, especially for response times, there is a big difference between goals and requirements (the point when stakeholders agree that the system can't go into production with such performance). For many interactive web applications, response time goals are two to five seconds and requirements may be somewhere between eight seconds and a minute.

One approach may be to define both goals and requirements. The problem is that, except when coming from legal or contractual obligation, requirements are very difficult to get. Even if stakeholders define performance requirements, quite often, when it comes to the go/no go decision, it becomes clear that it was not the real requirements, but rather second-tier goals.

In addition, multiple performance metrics only together provide the full picture. For example, you may have a 10-second requirement and you get 15-second response time under the full load. But what if you know that this full load is the high load on the busiest day of year, that response times for the maximal load for other days are below 10 seconds, and you see that it is CPU-constrained and may be fixed by a hardware upgrade? Real response time requirements are so environment and business dependent that for many applications it may be problematic to force people to make hard decisions in advance for each possible combination of circumstances. One approach may be to specify goals (making sure that they make sense) and only then, if they are not met, make the decision what to do with all the information available.

Determining what specific performance requirements are is another large topic that is difficult to formalize. Consider the approach suggested by Peter Sevcik for finding T, the threshold between satisfied and tolerating users. T is the main parameter of the Apdex (Application Performance Index) methodology, providing a single metric of user satisfaction with the performance of enterprise applications. Peter Sevcik defined ten different methods [3]

1. Default value (the Apdex methodology suggest 4 sec)
2. Empirical data
3. User behavior model (number of elements viewed / task repetitiveness)
4. Outside references
5. Observing the user
6. Controlled performance experiment
7. Best time multiple
8. Find frustration threshold F first and calculate T from F (the Apdex methodology assumes that F = 4T)
9. Interview stakeholders
10. Mathematical inflection point

Each method is discussed in detail in *Using Apdex to Manage Performance.*

The idea is the use of several of these methods for the same system. If all come to approximately the same number, they give us T. While this approach was developed for production monitoring, there is definitely a strong correlation between T and the response time goal (having all users satisfied sounds like a pretty good goal), and between F and the response time requirement. So the approach probably can be used for getting response time requirements with minimal modifications. While some specific assumptions like four seconds for default or the F=4T relationship may be up for argument, the approach itself conveys the important message that there are many ways to determine a specific performance requirement and it would be better to get it from several sources for validation purposes. Depending on your system, you can determine which methods from the above list (or maybe some others) are applicable, calculate the metrics and determine your requirements.

Usability requirements, mainly related to response times, are based on the basic principles of human-computer interaction. Many researchers agree that users lose focus if response times are more than 8 to 10 seconds and that response times should be 2 to 5 seconds for maximum productivity. These usability considerations may influence design choices (such as using several web pages instead of one). In some cases, usability requirements are linked closely to business requirements; for example, make sure that your system's response times are not worse than response times of similar or competitor systems.

"Usability requirements, mainly related to response times, are based on the basic principles of human-computer interaction."

AlexPODELKO

As long ago as 1968, Robert Miller's paper *Response Time in Man-Computer Conversational Transactions* described three threshold levels of human attention[4]. Jakob Nielsen believes that Miller's guidelines are fundamental for human-computer interaction, so they are still valid and not likely to change with whatever technology comes next[5]. These three thresholds are:

1. Users view response time as instantaneous (0.1-0.2 second)
2. Users feel they are interacting freely with the information (1-5 seconds)
3. Users are focused on the dialog (5-10 seconds)

Users view response time as instantaneous (0.1-0.2 second): Users feel that they directly manipulate objects in the user interface. For example, the time from the moment the user selects a column in a table until that column highlights or the time between typing a symbol and its appearance on the screen. Robert Miller reported that threshold as 0.1 seconds. According to Peter Bickford 0.2 second forms the mental boundary between events that seem to happen together and those that appear as echoes of each other[6].

Although it is a quite important threshold, it is often beyond the reach of application developers. That kind of interaction is provided by operating system, browser, or interface libraries, and usually happens on the client side, without interaction with servers (except for dumb terminals, that is rather an exception for business systems today).

Users feel they are interacting freely with the information (1-5 seconds): They notice the delay, but feel the computer is "working" on the command. The user's flow of thought stays uninterrupted. Robert Miller reported this threshold as one-two seconds[4].

Peter Sevcik identified two key factors impacting this threshold[7]: the number of elements viewed and the repetitiveness of the task. The number of elements viewed is, for example, the number of items, fields, or paragraphs the user looks at. The amount of time the user is willing to wait appears to be a function of the perceived complexity of the request. Impacting thresholds are the complexity of the user interface and the number of elements on the screen. Back in 1960s through 1980s the terminal interface was rather simple and a typical task was data entry, often one element at a time. Earlier researchers reported that one to two seconds was the threshold to keep maximal productivity. Modern complex user interfaces with many elements may have higher response times without adversely impacting user productivity. Users also interact with applications at a certain pace depending on how repetitive each task is. Some are highly repetitive; others require the user to think and make choices before proceeding to the next screen. The more repetitive the task is the better the expected response time.

That is the threshold that gives us response time usability goals for most user-interactive applications. Response times above this threshold degrade productivity. Exact numbers depend on many difficult-to-formalize factors, such as the number and types of elements viewed or repetitiveness of the task, but a goal of two to five seconds is reasonable for most typical business applications.

There are researchers who suggest that response time expectations increase with time. Forrester research of 2009 suggests two second response time; in 2006 similar research suggested four seconds (both research efforts were sponsored by Akamai, a provider of web accelerating solutions).[8] While the trend probably exists, the approach of this research was often questioned because they just asked users. It is known that user perception of time may be misleading. Also, as mentioned earlier, response time expectations depends on the number of elements viewed, the repetitiveness of the task, user assumptions of what the system is doing, and UI showing the status. Stating standard without specification of what page we are talking about may be overgeneralization.

Users are focused on the dialog (5-10 seconds). They keep their attention on the task. Robert Miller reported threshold as 10 seconds[4]. Users will probably need to reorient themselves when they return to the task after a delay above this threshold, so productivity suffers.

Peter Bickford investigated user reactions when, after 27 almost instantaneous responses, there was a two-minute wait loop for the 28th time for the same operation. It took only 8.5 seconds for half the subjects to either walk out or hit the reboot[6]. Switching to a watch cursor during the wait delayed the subject's departure for about 20 seconds. An animated watch cursor was good for more than a minute, and a progress bar kept users waiting until the end. Bickford's results were widely used for setting response times requirements for web applications.

That is the threshold that gives us response time usability requirements for most user-interactive applications. Response times above this threshold cause users to lose focus and lead to frustration. Exact numbers vary significantly depending on the interface used, but it looks like response times should not be more than eight to 10 seconds in most cases. Still, the threshold shouldn't be applied blindly; in many cases, significantly higher response times may be acceptable when appropriate user interface is implemented to alleviate the problem.

Analysis and Specification

The third category, technological requirements, comes from chosen design and used technology. Some technological requirements may be known from the beginning if some design elements are given, but others are derived from business and usability requirements throughout the design process and depend on the chosen design.

For example, if we need to call ten web services sequentially to show the web page with a three-second response time, the sum of response times of each web service, the time to create the web page, transfer it through the network and render it in a browser should be below 3 second. That may be translated into response time requirements of 200-250 milliseconds for each web service. The more we know, the more accurately we can apportion overall response time to web services.

Another example of technological requirements is resource consumption requirements. For example, CPU and memory utilization should be below 70% for the chosen hardware configuration.

Business requirements should be elaborated during design and development, and merge together with usability and technological requirements into the final performance requirements, which can be verified during testing and monitored in production. The main reason why we separate these categories is to understand where the requirement comes from. Is it a fundamental business requirement so that if the system fails we will miss it or is it a result of a design decision that may be changed if necessary.

Requirement engineering/architect's vocabulary is very different from what is used in performance testing or capacity planning. Performance and scalability are often referred as examples of quality attributes (QA), a part of nonfunctional requirements (NFR).

In addition to specifying requirements in plain text, there are multiple approaches to formalize documenting of requirements. For example, Quality Attribute Scenarios by The Carnegie Mellon Software Engineering Institute (SEI) or Planguage (Planning Language) introduced by Tom Gilb.

After 27 almost instantaneous responses, *there was a two-minute wait loop for the* 28th time for the same operation.

QA scenario defines source, stimulus, environment, artifact, response, and response measure[9]. For example, the scenario may be that users initiate 1,000 transactions per minute stochastically under normal operations, and these transactions are processed with an average latency of two seconds.

For this example:

- Source is a collection of users
- Stimulus is the stochastic initiation of 1,000 transactions per minute
- Artifact is always the system's services
- Environment is the system state, normal mode in our example
- Response is processing the transactions
- Response measure is the time it takes to process the arriving events (an average latency of two seconds in our example)

Planguage (Planning language) was suggested by Tom Gilb and may work better for quantifying quality requirements[10]. Planguage keywords include:

- **Tag:** a unique identifier
- **Gist:** a short description
- **Stakeholder:** a party materially affected by the requirement
- **Scale:** the scale of measure used to quantify the statement
- **Meter:** the process or device used to establish location on a Scale
- **Must:** the minimum level required to avoid failure
- **Plan:** the level at which good success can be claimed
- **Stretch:** a stretch goal if everything goes perfectly
- **Wish:** a desirable level of achievement that may not be attainable through available means
- **Past:** an expression of previous results for comparison
- **Trend:** an historical range or extrapolation of data
- **Record:** the best-known achievement

It is very interesting that Planguage defines four levels for each requirement: minimum, plan, stretch, and wish.

Another question is how to specify response time requirements or goals. Individual transaction response times vary, so aggregate values should be used. For example, such metrics as average, maximum, different kinds of percentiles, or median. The problem is that whatever aggregate value you use, you lose some information.

Percentiles are more typical in SLAs (Service Level Agreements). For example, 99.5 percent of all transactions should have a response time less than five seconds. While that may be sufficient for most systems, it doesn't answer all questions. What happens with the remaining 0.5 percent? Do these 0.5 percent of transactions finish in six to seven seconds or do all of them timeout? You may need to specify a combination of requirements. For example, 80 percent below four seconds, 99.5 percent below six seconds, and 99.9 percent below 15 seconds (especially if we know that the difference in performance is defined by distribution of underlying data). Other examples may be average four seconds and maximal 12 seconds, or average four seconds and 99 percent below 10 seconds.

Moreover, there are different viewpoints for performance data that need to be provided for different audiences. You need different metrics for management, engineering, operations, and quality assurance. For operations and management percentiles may work best. If you do performance tuning and want to compare two different runs, average may be a better metric to see the trend. For design and development you may need to provide more detailed metrics; for example, if the order processing time depends on the number of items in the order, it may be separate response time metrics for one to two, three to 10, 10 to 50, and more than 50 items.

Often different tools are used to provide performance information to different audiences; they present information in a different way and may measure different metrics. For example, load testing tools and active monitoring tools provide metrics for the used synthetic workload that may differ significantly from the actual production load. This becomes a real issue if you want to implement some kind of process, such as ITIL Continual Service Improvement or Six Sigma, to keep performance under control throughout the whole system lifecycle.

Things get more complicated when there are many different types of transactions, but a combination of percentile-based performance and availability metrics usually works in production for most interactive systems. While more sophisticated metrics may be necessary for some systems, in most cases they make the process overcomplicated and results difficult to analyze.

There are efforts to make an objective user satisfaction metric. For example, Apdex (application performance index) is a single metric of user satisfaction with the performance of enterprise applications. The Apdex metric is a number between zero and one, where zero means that no users were satisfied, and one means all users were satisfied. The approach introduces three groups of users: satisfied, tolerating, and frustrated. Two major parameters are introduced: threshold response times between satisfied and tolerating users T, and between tolerating and frustrated users F. There probably is a relationship between T and the response time goal, and between F and the response time requirement. However, while Apdex may be a good metric for management and operations, it is probably too high-level for engineering.

Validation and Verification

Requirements validation is making sure that requirements are valid (although the term 'validation' is quite often used to mean checking against test results instead of verification). A good way to validate a requirement is to get it from different independent sources; If all numbers are about the same, it is a good indication that the requirement is probably valid. Validation may include, for example, reviews, modeling, and prototyping. Requirements process is iterative by nature and requirements may change with time, so to be able to validate them is important to trace requirements back to their source.

Requirements verification is checking if the system performs according to the requirements. To make meaningful comparison, both the requirements and results should use the same metrics. One consideration here is that load testing tools and many monitoring tools measure only server and network time. While end user response times, which business is interested in and usually assumed in performance requirements, may differ significantly, especially for rich web clients or thick clients due to client-side processing and browser rendering. Verification should be done using load testing results as well as during ongoing production monitoring. Checking production monitoring results against requirements and load testing results is also a way to validate that load testing was done properly.

Requirement verification presents another subtle issue which is how to differentiate performance issues from functional bugs exposed under load. Often, additional investigation is required before you can determine the cause of your observed results. Small anomalies from expected behavior are often signs of bigger problems, and you should at least figure out *why* you get them.

When 99 percent of your response times are three to five seconds (with the requirement of five seconds) and 1 percent of your response times are five to eight seconds it usually is not a problem. But it probably should be investigated if this 1 percent fail or have strangely high response times (for example, more than 30 sec) in an unrestricted, isolated test environment. This is not due to some kind of artificial requirement, but is an indication of an anomaly in system behavior or test configuration. This situation often is analyzed from a requirements point of view, but it shouldn't be, at least until the reasons for that behavior become clear.

These two situations look similar, but are completely different in nature:

1. The system is missing a requirement, but results are consistent. This is a business decision, such as a cost vs. response time trade off.

2. Results are not consistent (while requirements can even be met). That may indicate a problem, but its scale isn't clear until investigated.

Unfortunately, this view is rarely shared by development teams too eager to finish the project, move it into production, and move on to the next project. Most developers are not very excited by the prospect of debugging code for small memory leaks or hunting for a rare error that is difficult to reproduce. So the development team becomes very creative in finding "explanations". For example, growing memory and periodic long-running transactions in Java are often explained as a garbage collection issue. That is false in most cases. Even in the few cases, when it is true, it makes sense to tune garbage collection and prove that the problem went away.

Another typical situation is getting some transactions failed during performance testing. It may still satisfy performance requirements, which, for example, state that 99% of transactions should be below X seconds – and the share of failed transaction is less than 1 percent. While this requirement definitely makes sense in production where we may have network and hardware failures, it is not clear why we get failed transactions during the performance test if it was run in a controlled environment and no system failures were observed. It may be a bug exposed under load or a functional problem for some combination of data.

When some transactions fail under load or have very long response times in the controlled environment and we don't know why, we have one or more problems. When we have unknown problems, why not track it down and fix in the controlled environment? It would be much more difficult in production. What if these few failed transactions are a view page for your largest customer and you won't be able to create any order for this customer until the problem is fixed? In functional testing, as soon as you find a problem, you usually can figure out how serious it is. This is not the case for performance testing: usually you have no idea what caused the observed symptoms and how serious it is, and quite often the original explanations turn out to be wrong.

Michael Bolton described this situation concisely[11]:

As Richard Feynman said in his appendix to the Rogers Commission Report on the Challenger space shuttle accident, when something is not what the design expected, it's a warning that something is wrong. "The equipment is not operating as expected, and therefore there is a danger that it can operate with even wider deviations in this unexpected and not thoroughly understood way. The fact that this danger did not lead to a catastrophe before is no guarantee that it will not the next time, unless it is completely understood." When a system is in an unpredicted state, it's also in an unpredictable state.

To summarize, we need to specify performance requirements at the beginning of any project for design and development (and, of course, reuse them during performance testing and production monitoring). While performance requirements are often not perfect, forcing stakeholders just to think about performance increases the chances of project success.

What exactly should be specified – goal vs. requirements (or both), average vs. percentile vs. APDEX, etc. – depends on the system and environment. Whatever it is, it should be something quantitative and measurable in the end. Making requirements too complicated may hurt. We need to find meaningful goals / requirements, not invent something just to satisfy a bureaucratic process.

If we define a performance goal as a point of reference, we can use it throughout the whole development cycle and testing process and track our progress from the performance engineering point of view. Tracking this metric in production will give us valuable feedback that can be used for future system releases.

REFERENCES

[1] Guide to the Software Engineering Body of Knowledge (SWEBOK). IEEE, 2004. http://www.computer.org/portal/web/swebok

[2] Barber, S. Get performance requirements right - think like a user, Compuware white paper, 2007. http://www.perftestplus.com/resources/requirements_with_compuware.pdf

[3] Sevcik, P. Using Apdex to Manage Performance, CMG, 2008. http://www.apdex.org/documents/Session318.0Sevcik.pdf

[4] Miller, R. B. Response time in user-system conversational transactions, In Proceedings of the AFIPS Fall Joint Computer Conference, 33, 1968, 267-277.

[5] Nielsen J. Response Times: The Three Important Limits, Excerpt from Chapter 5 of Usability Engineering, 1994. http://www.useit.com/papers/responsetime.html

[6] Bickford P. Worth the Wait? Human Interface Online, View Source, 10/1997. http://web.archive.org/web/20040913083444/http://developer.netscape.com/viewsource/bickford_wait.htm

[7] Sevcik, P. How Fast Is Fast Enough, Business Communications Review, March 2003, 8–9. http://www.bcr.com/architecture/network_forecasts%10sevcik/how_fast_is_fast_enough?_20030315225.htm

[8] eCommerce Web Site Performance Today. Forrester Consulting on behalf of Akamai Technologies, 2009. http://www.akamai.com/html/about/press/releases/2009/press_091409.html

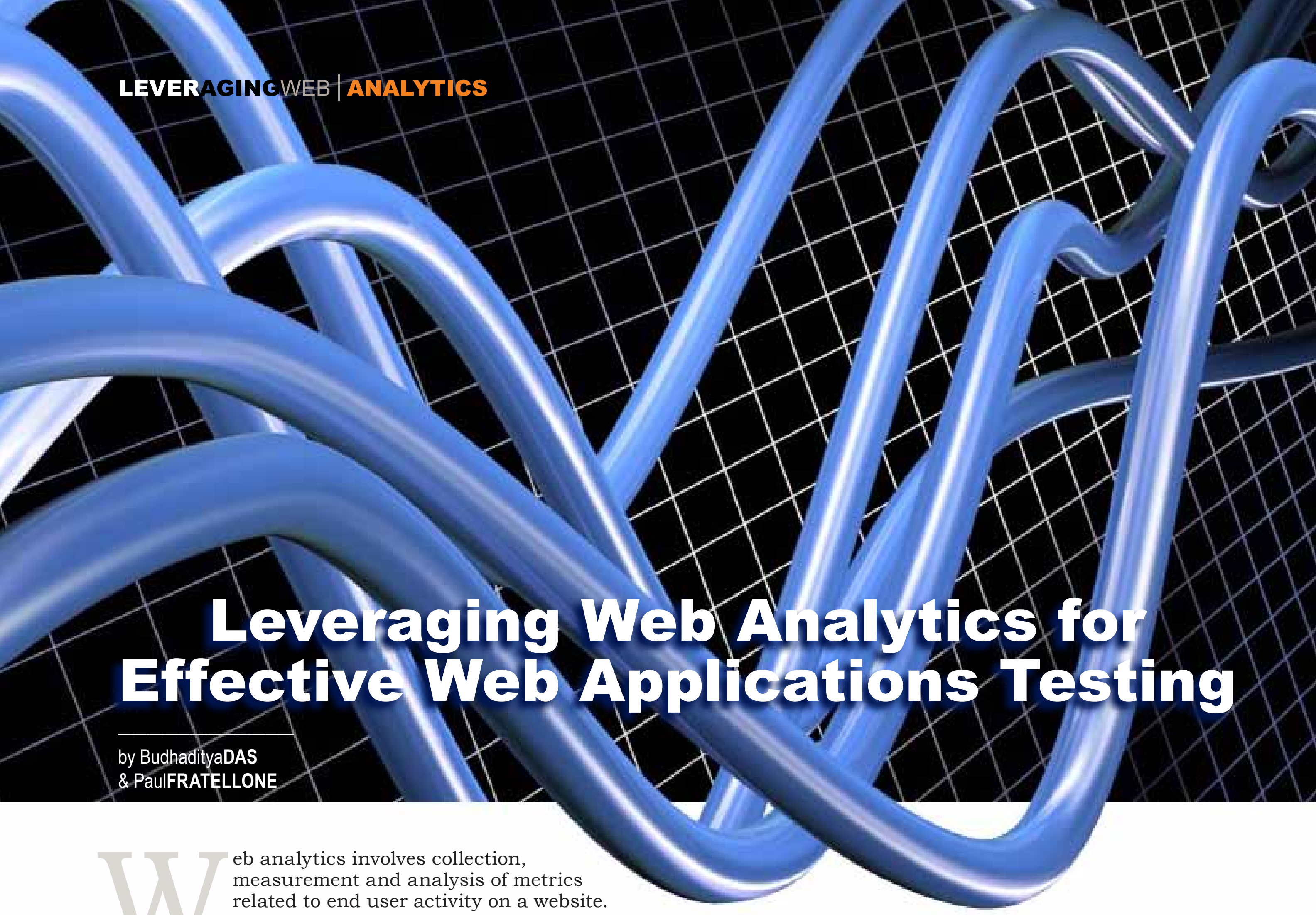
[9] Bass L., Clements P., Kazman R. Software Architecture in Practice, Addison-Wesley, 2003. http://etutorials.org/Programming/Software+architecture+in+practice,+second+edition

[10] Simmons E. Quantifying Quality Requirements Using Planguage, Quality Week, 2001. http://www.clearspecs.com/downloads/ClearSpecs20V01_Quantifying%2 Quality%20 Requirements.pdf

[11] Bolton M. More Stress, Less Distress, Better Software, November 2006. http://www.stickyminds.com/sitewide.asp?ObjectId=11536&Function=edetail&ObjectType=ART

About The Author

__Alex Podelko__ has specialized in performance engineering for the last fourteen years. Currently he is Consulting Member of Technical Staff at Oracle, responsible for performance testing and tuning of Hyperion products. Alex has more than 20 years of overall IT experience and holds a PhD in Computer Science from Gubkin University and an MBA from Bellevue University. Alex serves as a board director for Computer Measurement Group (CMG), His collection of performance-related links and documents can be found at http://www.alexanderpodelko.com/



Leveraging Web Analytics for Effective Web Applications Testing

by BudhadityaDAS
& PaulFRATELLONE

Web analytics involves collection, measurement and analysis of metrics related to end user activity on a website. Modern web analytics systems like Omniture, Tealeaf, Google Analytics and Webtrends offer powerful tools and infrastructure for measuring and analyzing website traffic patterns as well as usage trends for business and market research. These web analytics systems provide data on various traffic and end user statistics including number of visitors, page views, average session duration, popular pages, and common user workflows / click paths. Historically, these metrics were primarily utilized for market and business research.

Web analytics systems have grown fairly sophisticated over the last decade. The current generation systems can capture more granular and finer statistics for accurately "visualizing" end user behavior patterns and interaction with a web site. These detailed metrics are extremely useful for proactive web application optimization. Hence, web analytics systems are increasingly being deployed and utilized by business as well as technical stakeholders for managing, measuring, analyzing and enhancing end user experience.

The power of web analytics can be used by testing teams for understanding end user behavior on the web application under review. If effectively utilized, these statistics can help the test team design robust, accurate and efficient business workflows that mimic end user experiences for testing purposes. In this article, we endeavor to highlight how web analytics metrics and statistics can be effectively leveraged for optimizing and refining web application testing strategy.

Introduction

The software application delivery paradigm has undergone enormous change over the last few decades. There has been a steady march towards a web-centric model. This mass migration, away from the traditional client server model to rich internet-based applications, has been steadily increasing since the turn of the century.

The Internet era was ushered in with a bang. By the late 90's, thanks to the steady commercial growth of the Internet and the ubiquity of web browsers, there was an exponential reduction in the costs associated with personal computing. It is a different story that the proverbial 'bust' soon followed the 'bang'. But the ability to deploy, update and maintain applications without distributing and installing software on potentially millions of client computers and the inherent support for cross-platform compatibility forever changed the dynamics of software delivery ecosystem. By early 2000, the steady march towards a web-centric model turned into a deluge and not only enterprise businesses but also consumer applications (web mail, retail sales, etc.) started adopting this model with great fervor.

The latest buzzwords, "Software-As-A-Service (SAAS)" and "Cloud Computing" are a logical extension of the web-centric software delivery paradigm. From Office Productivity Suites (Google Docs, MS Office Live, Zoho Office) to CRMs (SalesForce.Com), on-demand web applications appear to be latest mantra.

Web Application Testing Landscape

On one hand the mass migration to the live, web-based delivery model is a boon for application vendors as it opens up a larger revenue stream. But the side effect is that web applications have to be always ready, functionally robust and scalable to sustain the ever-increasing demands of the user base while catering to positive user experience. A bad user experience can negatively affect revenue in both the short and the long term, not to mention the bad press that can be generated through social media sites. Missed opportunities and the loss of future revenue streams can be a deadly combination to the survival of the business.

As the world continues to embrace increasingly feature rich web-based products, technology delivery organizations must realize how to best help a business in meeting its goals and objectives. One of the most important aspects of this process is to ensure a positive experience for the end user through exhaustive validation of various quality dimensions of reliability, security, accuracy, performance, conformance and usability. This level of validation can only be achieved through comprehensive testing and quality assurance of web applications.

1.1 Web Application Testing Challenges

Modern web applications are complex systems, providing hyper-textual contents, computational facilities and services and can cater to diverse and large audiences. Correspondingly, the quality of web applications is a complex, multidimensional attribute. To ensure that they work reliably and correctly under different situations, multiple factors need to be accounted for and tested.

In addition to the routine challenges associated with application testing, web application testing introduces some additional nuances. The following section highlights some of the challenges associated with modern web application testing:

- **Large number of application usage workflows:** The hyper-contextual nature of web applications lends itself to a potentially large number of workflows for even simple scenarios. Testing all potential workflow permutations and combinations can be fairly complex. The challenges are further aggravated in the case of large and complex web applications
- **Large and diverse user base:** Web applications introduce various dimensions in the overall landscape. Not knowing the user type or any information about them, the situation is compounded by not knowing hardware and software configurations, network/ISP, connectivity speeds and browser types. This list is quite comprehensive and the testing organization must fully understand how this will affect the complexity, the number/permutations and effort/cost in attaining the level of quality the business owner expects. Hence, the testing team needs to ensure that all the aforementioned dimensions are accounted for as part of the overall testing process. These combinations can exponentially blow up and end up being time and cost prohibitive.
- **Diverse test infrastructure/environment:** Testing the aforementioned dimensions requires investment in various test infrastructure setups. This adds additional complexity in the overall process of web application testing.
- **Non-functional testing needs:** Because of the nature of a web-based application delivery model, quality metrics related to scalability and security gain special importance. Attention needs to be paid to security and information assets. There can be strict regulations and non-compliance can be a very costly mistake. Hence, performance and security testing become extremely critical elements of the overall testing dimensions.

In essence, the various dimensions associated with the web-based delivery model significantly increase the efforts required for exhaustive testing. With increasing time-to-market pressures and shrinking release schedules, the critical aspect of successful web application testing strategy boils down to prioritizing the efforts based on various factors. This is where web Analytics can play a key role.

The following sections provide an introduction to web analytics and how it can be leveraged to enhance web application testing efficiency.

Web Analytics 101

Most of us have encountered the omnipresent counter/hit counter/visitor counter on some web sites. This is web analytics in its simplest form.

A more formal definition of web analytics is as follows: “Web analytics is the measurement, collection, analysis and reporting of internet data for purposes of understanding and optimizing web usage.[1]” Web Analytics involves gathering various metrics related to web Application usage, in order to effectively uncover information related to various usage statistics. Some common questions that web analytics helps uncover include:

- Information about unique visitors, total visitors, page views, hits, etc.
- Statistics about the most accessed pages, workflows and images.
- Statistics about average (min/max/percentile) number of visitors, concurrency over a period of time and most busy time of day/week for traffic.
- Details about average error rates across various pages and types of errors.
- Various types of users and user profiles.
- Most common/least common user workflows and exit pages.
- Traffic patterns and trends, average session duration of users; in simple words the lengths of time users spend on a web site.
- Statistics related to geographical distribution of users.

Early web analytics techniques dealt with the analysis of web server logs using tools (known as web log analysis software) in order to extract various usage metrics. This is known as off-site or server side data collection. Web analytics systems have grown fairly sophisticated over the last decade and the current generation systems can

capture more granular and finer statistics for accurately visualizing end user behavior patterns and interaction with a website. This is achieved with the help of additional methods like JavaScript Page Tagging[2], Cookie Tracking, and IP Geo Location Tracking. Additional techniques involve the use of an Invisible Image[3] to implement a call back to the server from the rendered page. All the aforementioned techniques are collectively known as on-site (or client-side) web analytics.

Modern web analytics systems providers like Omniture, Tealeaf, Google Analytics, and Webtrends primarily utilize on-site analytics. Some of them even offer hybrid methods that combine on-site as well as off-site analytics for even more detailed metrics collection and analysis. The following lists out some of the common web analytics metrics:

Common Web Analytics Metrics

- *Unique Browsers/Browser Types*
- *Number of New Visitors*
- *Visits/Unique Visitors, Unique Authenticated Visitors*
- *Average Number of Visits per Visitor*
- *Page Views/Exits or Site Exits*
- *Average Number of Page Views per Visit*
- *Clicks/Impressions/Hits*
- *Average Pages Viewed per Visitor*
- *Page Views per Visit*
- *Ratio of New to Returning Visitors*
- *Visits per Month (or quarter or week)*
- *Conversion of non-subscribers to Subscribers*
- *Page Views per Visit-Active Subscriber Base*
- *Visit Duration/Session Duration*
- *Average Subscription Length*

Metrics collection is one part of the story. Most modern web analytics toolkits enable the correlation of these metrics with various business and financial figures and statistics using a range of data mining/BI techniques and algorithms. This detailed correlated information is extremely useful for proactive web application optimization. Hence, web analytics systems are increasingly being utilized by business as well as technical stakeholders for managing, measuring, analyzing and enhancing end user experience.

Leveraging Web Analytics for Effective Testing Strategy

The adage "information is power" is an understatement in terms of the value the analytics provide to the business and technology teams. For QA/testing, we can use this information to validate our user cases or business scenarios, test case development techniques, test case coverage and depth of testing, regression testing and user acceptance testing.

1.2 Leveraging Web Analytics for Functional Testing

- **Prioritizing the business workflows / test scenarios based on analytics metrics:** Using various analytics metrics, the ability for the business to prioritize can be affirmed. Testers benefit from our ability to extract, tag and execute (test cases) the most common paths and configurations, making for an easy exercise with a high degree of confidence. Understanding utilization patterns of the users in how they navigate the site will also ensure that the most traversed paths are tested before the less common ones. You cannot test every combination or path – there is the law of diminishing returns – but ensuring your tests cover the most important and used features, under the most common configurations should enable a level of success for the business.
- **Enhancing test scenarios by extracting unique workflows:** Most modern web analytics tools provide detailed information about various workflows executed by users on the application. These statistics can be a rich source of information for extracting unique workflows and work paths that may have been missed during the initial test design process.
- **Optimizing test environment:** Web analytics can provide a lot of useful metrics related to browser, OS, and network speed associated with the applications' diverse user base. This information can be effectively utilized for prioritizing the environment setup needs.

1.3 Web Analytics for Performance Testing

One of the biggest challenges in effective performance testing is to ensure the simulated load pattern is realistic and mimics real life traffic patterns on the web application.

Metrics gathered from web analytics tools can be very useful in defining and designing realistic load models.

- **Identifying Load Distribution:** Metrics gathered from web analytic tools are a really good source of information for understanding the various load levels (average/ min/max) and traffic trends on the web application.
- **Workload Distribution:** A workload distribution is a representation of the functions performed by a user community on the application. For example, during the course of a day on a retail-based website, most users are shopping, some are doing a search for a specific product, some are checking out and all while a single administrator may be updating prices on products. Analysis of various web analytics metrics (page, visit and user-related statistics) can provide suitable insight for designing realistic workload models. The following figure shows a sample "Workload Model" for a video on demand application.

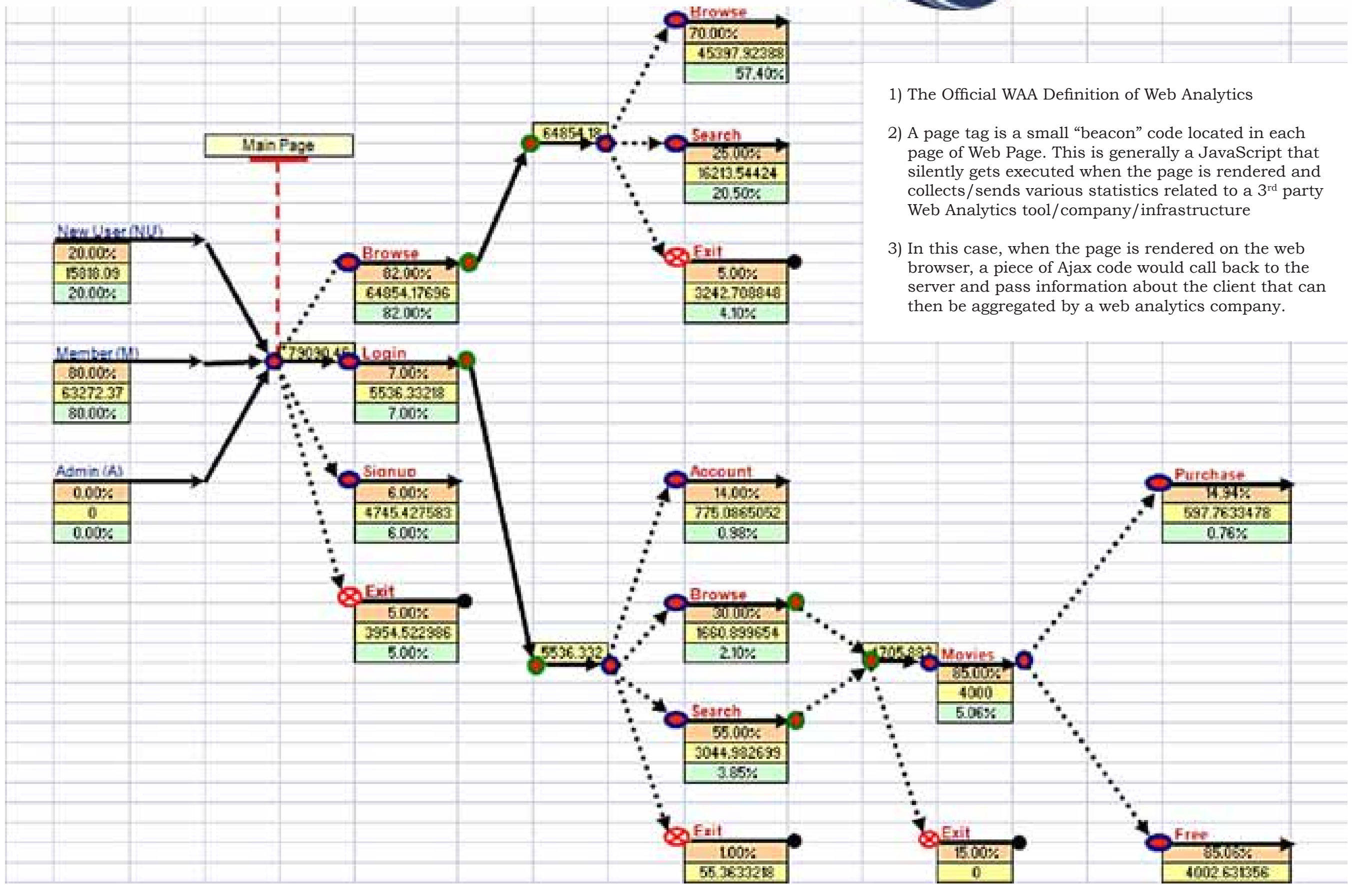


Figure 1: *Workload Model for a Video on Demand Web Application*

1) The Official WAA Definition of Web Analytics

2) A page tag is a small "beacon" code located in each page of Web Page. This is generally a JavaScript that silently gets executed when the page is rendered and collects/sends various statistics related to a 3rd party Web Analytics tool/company/infrastructure

3) In this case, when the page is rendered on the web browser, a piece of Ajax code would call back to the server and pass information about the client that can then be aggregated by a web analytics company.

Web analytic metrics can be very useful for enhancing the testing efforts of web applications. Feedback from analytics can be continually evaluated against current testing to ascertain if there is a need to make changes to tests in regression and user acceptance.

Analytics will guard against testers doing less valuable exploratory testing on areas less frequently used or having no real business impact.

Conclusion

Visitors are driven to websites for many reasons. They interact with the site in many ways and whether we plan for it or not, they navigate the site motivated by the context of their reasons and their experience will affect their decisions as consumers. Competition is everywhere and someone else is guaranteed to offer what you do.

As testers, one needs to make sure that the information we use to ensure a satisfactory level of testing has been achieved and must be quantified and measurable. Analytics supports the test strategy and plan in such a manner. When testing costs are challenged, and/or time constraints limit our ability to deliver a product of high quality, one can turn to web analytics and clearly quantify the value of testing and the risk of not testing.

At the end of the day, effective analysis of web analytic metrics can provide deep insight into the overall web application usage landscape. If effectively leveraged, this can be a gold mine of information for deducing, reducing and prioritizing overall testing efforts.

About The Author

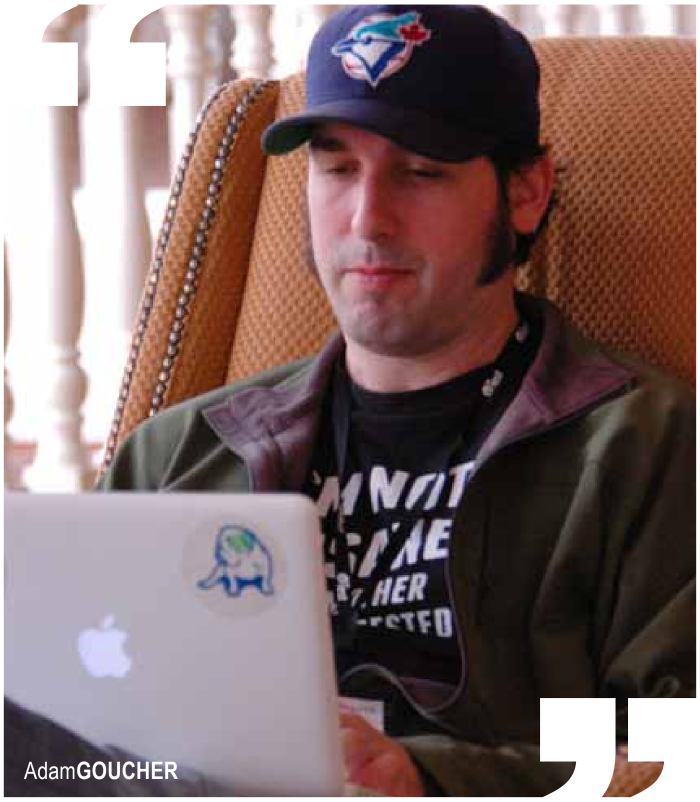
Budhaditya Das *is a Performance and Automation Specialist in MindTree's Test Consulting Group. He has more than 7 years of product testing experience with expertise in performance testing, test automation and security testing.*

Paul Fratellone *has a 25-year career in information technology centralized in Testing & Quality Management. Performing every activity/task of the QA professional from QA Analyst to Director, he has experienced many challenges and opportunities to deliver quality through testing.*

ASK THE TESTER

Compiled By Matthew HEUSSER

Matt contributes to the STP community blog at http://www.softwaretestpro.com/blog or follow him on Twitter @mheusser


AdamGOUCHER

He tests software, writes automation (java, selenium, python, ruby, take your pick) and yes, used to coach Lacrosse. Of course I'm talking about Adam Goucher, the Canadian Test Consultant and Contractor.

Beyond test contracting, Adam is probably best known as a writer. A co-editor of *Beautiful Testing* (2009, O'Reilly), Adam is a prolific blogger; he also tweets as @AdamGoucher.

If I had to pin his skills down, I'd put Adam in the school of context-driven testing (hey, that's what he says on his about page on his website) but also with a focus on automation and enabling developers to create very complex test setups, rigs, harnesses, etc.

In fact, in my words (not his) I would say Adam is the guy people call when they want to do something tricky with test automation and get stuck.

Adam Goucher will now answer your questions.

Adam's blog: http://adam.goucher.ca/

Beautiful Testing is available wherever fine books are sold.

Next issue we'll interview Mike Lyles, the QA Program Manager at Lowe's Companies, Incorporated. Please email your questions, name and location with 'Ask the tester' as subject line to ***matt.heusser@gmail.com***

QUESTION *Lately our company has been purchasing applications that are client-based which I get asked to performance test. We use IBM's Rational Performance Tester, which we use primarily for web-based applications and web services. I've also got access to Visual Studio Team System Test Edition, but the training material states you can only use one virtual user to drive GUI tests. Is driving traffic through the GUI's something that is done routinely using a different set of tooling or is this a stumbling block for everyone? The idea of purchasing yet another expensive product won't go over well so I'm curious what others are doing.*

– Michael Beckman

GOUCHER There are currently at least two approaches to doing performance testing of applications through a browser that I use. Keeping in mind I am aligned with Selenium, they both involve it.

Approach one – death by a billion browsers: Using BrowserMob, I can unleash an army of browsers (controlled by Selenium) to assault the site. Since it is SaaS, I don't need to setup or maintain the load generation machines and I only pay for what I need. It allows me to setup different load profiles to execute concurrently which lets me simulate the real (or expected) traffic patterns.

Approach two – you are the weakest link, good-bye: Using some tool, apply pressure on a key resource (network, cpu, memory) to simulate a constrained situation and then hit the server with a handful of requests. You can then see what goes POP by moving the constraint around from the load balancer, to the app tier, to the database, the caching server, etc.

Or combine both.

Notice how neither involve the build-out of your own massive infrastructure the way we might have had to do it ten years ago. Capital is too expensive and difficult to acquire today to waste it acquiring machines that just sit idle during the two months between performance runs (though the *huge* Sun machine we bought for it was pretty cool). There is still a monetary component, yes, but I would guess it is less than what you would spend on a tool for use in-house once you factor in annual maintenance contracts, hardware to run on, training, etc.

Full disclosure, I have relationship with BrowserMob as part of my consulting business.

QUESTION *Selenium seems to be having an identity crisis. The Selenium IDE is the public face of the project, yet very few active users use it (or admit to using it). What's the best way to get beyond the IDE, and help make the transition to using the real Selenium under the hood? Additionally, how can we help make that change occur with our testing friends?*

– Michael Larsen, *San Francisco, CA*

GOUCHER Ya. We kinda got called out on that by Bret Pettichord at the Selenium Conference. And I don't think we yet have a good answer to that. Part of it is education around what sort of things Se-IDE is good at (rapid bug reproduction scripts or driving the application to a known state to reduce startup time for exploratory testing are principle ones) and what the RC and WebDriver APIs allow for (data driving, integration with backend databases and web services, random input generators, sophisticated oracles, etc.).

But even that doesn't really cover it all. There is an additional education burden to learn how to program (well enough to automate) and to make effective use of the various xunit frameworks that exist. And not everyone wants to learn how to do this. A motivated learner can learn enough Python to be effective with Selenium and Page Objects in 3 days of hard work.

The trick there is motivated.

And not everyone is.

The key to a person's motivation is entirely unique to them, but you have to always be wary of not inflicting help upon them with automation. Some of the best testers I know would not list automation as their primary tool. And some of the worst I have seen do.

Circling back to the identity crisis, I suspect things are about to get a lot worse than better in the near term now that Selenium has commercial interests and their own marketing agendas. Not to mention what amounted to a declaration of war against HP at the conference. The ramifications of that could be interesting.

QUESTION *Follow-Up Question – How can we help our testing friends make the switch from IDE to a more robust Selenium?*

– Michael Larsen, *San Franciso, CA*

GOUCHER What we need is more turn key automation frameworks – along the same idea that things like QTP are turn key – that provide a structure around the automation, integrates the runner and are basically ready to go out of the box. Eventually, as the automation becomes more sophisticated then they will become hyper customized (see the 'Lightsaber Heuristic' below), but I have found that getting the initial bit of inertia is often the biggest challenge people face starting out with automation.

It is my hope that I'll be able to make an announcement on something to this end in the next month or so.

QUESTION *Hello Adam! At CAST 2009 you gave a talk,* The Most Important Stakeholder Is You *(http://adam.goucher.ca/papers/ TheMostImportantStakeholder- You.pdf). I'm curious, in what manner has the world changed since then? How has the testing community changed with respect to your statements?*

– Jeroen Rosink, *The Netherlands*

GOUCHER The ideas in that paper are still just as relevant today; only now, instead of "just" Twitter we have "Twitter the juggernaut." A lot of people, including myself, have reduced their blogging but instead discuss, spread and consume ideas and do a lot of our marketing through Twitter. One thing I didn't mention in the paper is the value in having your name as your brand. Very few can pull off having something clever as their online persona and have it map back to them the way Elizabeth Hendrickson can with "test obsessed" for example – but I'm sure she has put in the requisite effort into making that happen. Unless you are going to make that effort, then just stick with being you.

But always remember, that like any social media, your past, current or even future clients (employers) could also be watching you.

Oh, and fair warning, I have been told I get chatty on Twitter when at airports.

QUESTION *Adam, I have never worked with Selenium, although it looks interesting and I'd like to learn it. At STPCon, there were a group of testers sitting at a table next to mine who were talking about testing with Selenium. A heated point of discussion was whether Selenium would work for their environment or not. Without being any more rude than eavesdropping, which I was already doing, they peaked my interest with that. One fellow was adamant that there was no way that Selenium would help them. As I said, I am curious and have never worked with Selenium. Can you describe for me an environment or application that would be perfect for Selenium and one where Selenium would absolutely be of no value?*

– Pete Walen, *Grand Rapids, MI*

GOUCHER Selenium does web content. That's pretty much it. So if you have content in a browser then Selenium could be a perfect tool to interact with it.

There are exceptions. For example, if you are using Flash or Flex on your site then you will need to do extra work to allow Selenium to peer inside the black box that is the stage. But there are known solutions for that. Same for Silverlight or HTML 5's Canvas tag (which was wonderfully demonstrated at the Selenium Conference earlier this month – certainly a video to watch once they are released). Java Applets are another black box, though I have heard of one team making a bridge so they can use Selenium with it as well.

Certain web frameworks are also a pain to deal with due to random id generation, buttons that are not buttons but styled javascript, etc., but there are techniques for working around that as well. (And they likely are a pain even in commercial tools as well.)

And with the introduction of the WebDriver API into the Selenium mix, other problems that used to plague Selenium like upload file dialogs are also less of a problem.

Not to mention its cross-platform, multi-language support that other tools cannot (or do not currently) provide.

QUESTION *How would you best visually demonstrate the value of testing to a business person or someone who doesn't understand testing?*

– Lanette *"Dairy is Better than Non-dairy"* Creamer, *Seattle, WA*

GOUCHER Visually, I'm not sure. But as someone who has been hired a number of times to start testing groups from scratch, I take two approaches to conversations like this. (Both of which I got from Michael Bolton who in turn got them from Jerry Weinberg I believe.)

First, I need to understand *why* we are having this conversation in the first place. Why is it that now, after they have been doing things a certain way for a certain length of time, do they think they need to have testing. Understanding what event(s) led up to the conversation is important. With that information you now know the existence of a stakeholder, who they are and their motivations. Everything (including testing) is, after all, a people problem.

Second, I try to understand, or at least get them to explain what would happen without testing. Perhaps testing in their circumstance is exactly the wrong thing to do and is testing-for-the-sake-of-testing rather than an activity that has beneficial results. Perhaps spending a week on testing pushes their release date beyond an important industry event. Immediately after it being a people problem, it is a money problem.

As a consultant, I find a bigger challenge in a lot of organizations is When To Stop Testing rather than whether or not to do testing at all.

The Grand Rapids Airport Parking Lot Calculator is a fine example of this. Yes, there are massive, and fun, bugs in it – but it does work when used in the context it was designed for. More testing wouldn't add any further value to either the airport or people wanting to calculate parking charges, but would result in more bugs being found.

"As a consultant, I find a bigger challenge in a lot of organizations is When To Stop Testing rather than whether or not to do testing at all."

Adam**GOUCHER**

QUESTION *What's the all-time weirdest question you've ever been asked about Selenium?*

– Catherine Powell, *Bahston, MA*

GOUCHER I'm not sure that I've had an all-time weirdest question; or at least nothing that pops out. But at least once a month I see someone trying to automate either GMail or Hotmail or similar which always makes me shake my head in disbelief. Unless you are Google (or Microsoft respectively) you should not be trying to automate those things. There is always a better solution to what you are trying to do. Usually it involves reaching into the database to get the authentication token that was sent to the user's GMail account and faking out the browser into thinking the user clicked the message – but sometimes it is just easier not to automate that particular task.

On the cool side of things instead, Chris McMahon won an iPad for crawling Craigslist and I have heard of people checking into their flights via Selenium as well. I also have got into the habit of very carefully reading the rules of online contests for entries via automated means and entry limits by email and/or household.

QUESTION *Imagine that you have a newbie tester who dreams to become the automation tester one day, what would be your advice for him and what steps should he take ?*

– Aleksander Lipski, *Poznan, Poland*

GOUCHER I'm pretty sure David Burns would have issue with someone else wanting that title, but

Step one: learn to test

Step two: learn to program

I think it actually should be in that order as well. One thing I am learning is that being a tester-that-automates is very different than a developer-that-automates. And there are too many of the latter and not enough of us as the former.

Automation is all about looking for the oracle and figuring out what switches and levers need to be pulled in order to meet your mission. It is not just making things flash across the screen at super-human speed. But it is easy to not know that if you don't come from a testing background or have not become someone who identifies more as a tester than a developer.

As for what language to learn, since that is often the follow-up to this, I would say either Python or Ruby.

QUESTION *(Follow Up) What is your opinion regarding Page Object Pattern. Should we in general create automation scripts with this pattern or do you have another approach for dry and bulletproof automation ?*

– Aleksander Lipski, *Poznan, Poland*

GOUCHER To quote from a slide of mine: Page Objects FTW!

To me, the big benefit for the (Selenium) community so far has been the popularization of the Page Object Pattern which, for those not indoctrinated yet, is basically to represent pages under test as either a single class or group of classes. This brings the full power of OO to your automation framework. Done well, it leads to scripts that are extremely readable and completely removed from their implementation allowing for quick updates when locators change or even replacement of the underlying driver technology should the mood strike – without the script changing.

Yes, it is possible to write bulletproof and dry automation without it, but the separation between interface and implementation is a lot harder.

When I teach Selenium, I just start with Page Objects these days skipping the traditional way of writing scripts entirely.

QUESTION *You've written recently that the next leap in the testing field will come from another discipline. Apart from something that might help any technical worker in general, such as advancements in learning or introspective techniques, do you think that a technological or purely testing-related methodology will manifest itself? If so, will it be related to automation (i.e. the next Selenium) or will it be a new school of testing, or of software development? And, of course, Why?*

– Alex Kell, *Atlanta, GA*

GOUCHER Don't suppose I can just say 'No', can I?

I still think that we're pretty tapped out from looking inward. The best books, podcasts and conversation I have had on testing all had nothing to do with testing, at least directly. The ideas popularized in Malcolm Gladwell's *Blink* for instance go nicely with (my ideas on) exploratory testing for example. Or Sian Beilock's 'Choke' on how to deal with pressure (like, oh, say a pending product release). Or closer to the technology world, things like Continuous Delivery – which came out of more the IT Ops field than it did testing. And Agile before it came from the development side of the building.

I would be much happier if more testing conferences and magazines would have non-testers as speakers or authors. Their contribution is what will grow the field and the practice more than just the usual suspects.

QUESTION *As a mostly commercial tools shop, we look at the open source tools frequently so see what all the hype is about. We have a very stable and solid framework using BPT, or Business Process Testing – (approaching 2 years and 400 tests with a very manageable maintenance load). We can never see how the open source tools can even begin to cover all the bases that our BPT framework does on complex applications.*

All of our searches for published success stories detail extremely simple systems.

How do you implement these tools on more complex systems? Have you seen them succeed?

– Jason, *Pennsylvania, USA*

GOUCHER First, if you have a *very stable and solid framework* already then you would be silly for changing it just for the sake of change. Open source or otherwise.

Secondly, and this is where I become open source anarchist for a bit, what you end up with when you go the open source route is a solution that actually meets your needs rather than needs that meet your solution. For simple applications, the notion of cost can also be a factor, but in large organizations and complex systems even free solutions have a cost.

HP seems to have monopolized all search results for BPT so I'm guessing this is what you are talking about. From a cursory glance at their product page, it looks like it is in the same space as Cucumber or Robot Framework – only wrapped in a pretty GUI rather than being text.

(Though depending on how the assets are stored, the GUI could actually be a hindrance.) But I've not used HP-BPT before, or even heard of it until now, so my interpretation could be wrong.

In any event, any tool that promises non-programming automation is peddling snake-oil including the open source ones. Automation is programming. Sorry. But that is just the way it is.

Finally, building your own automation environment is something I call the 'Lightsaber Heuristic'. At one point, it was Star Wars canon that a Jedi needed to go on the Epic Quest to find the components to build their lightsaber. It after all was their weapon and tool that would one day save their life which meant they needed to understand it completely and 'be at one' with it. Same rules apply to automation – though hopefully it is not in a role where it can have your life in the balance. COTS tools are like getting your lightsaber from a vending machine in the mall; you don't know how they are built, you don't have access to spare parts, and even if you did have access to them, taking the back cover off voids the warranty.

QUESTION *What can Selenium do for the iOS automation? How far can we take iOS (next up android) testing?*

– Jane Fraser, *Bay Area, California*

GOUCHER Selenium 2, which has both the Remote Control and WebDriver APIs can do automation of IOS applications to some extent via WebDriver. It cannot drive all the native controls, which makes sense since Selenium is a browser automation library after all, but it can drive ones that use UIWebView (a WebKit based browser) both in the emulator and on an attached device. I suspect the information on the Selenium wiki might be a bit out of date, but I know it has been discussed on the webdriver mailing list a number of times.

QUESTION *How did you manage to get all the chapters for* Beautiful Testing? *It must have been like herding cats! What is the secret to getting multiple contributors to step up to the plate on a book such as that?*

– Lisa Crispin, *Denver, Colorado*

GOUCHER Lots, and lots of email. Unbelievable amounts of email actually. I did the total once and I think it was close to 2000 individual messages from the time we pitched the idea to O'Reilly to the time it was published.

In terms of getting contributors, it actually wasn't as hard as you might think and I suspect it was due largely by its association with Nothing But Nets. Contributors, and editors, agreed to donate all revenues to Nothing But Nets to purchase mosquito nets for Malaria infested regions. Every 45 seconds a child dies of Malaria, so donating time to write an article was almost a humanitarian effort. There were a few people that were approached that I'm disappointed didn't participate, but after a while you understand both how busy people can be, and the politics of the testing world.

My next book will be just me though. :)

QUESTION *Mercury Interactive is a former company that used to sell test tools. Is there any truth in the story that the Selenium name was chosen because Selenium is an antidote to Mercury poisoning?*

– Shmuel Gershon, Isreal

GOUCHER Yes, though it was a bit of an indirect route to get there. Jason Huggins, creator of Selenium, told the full story at his Selenium Conference keynote. If I recall correctly, his original name was to be CheckEngine to continue with Thoughtworks' car theme (they had CruiseControl already) but a conversation here and there led him down the road (another car metaphor!) to Selenium. His keynote was recorded so at some point in the future you'll be able to get the full story.

QUESTION *Which misconception about test automation is/are the most popular? Are there you have to correct or explain over and over again?*

– Shmuel Gershon, *Isreal*

GOUCHER The biggest one around Selenium in general is that somehow Selenium is going to magically determine that your site is wonderful. Selenium is simply a library for controlling a browser, nothing more. The ability to data drive scripts, do clever modeling of workflows, assertions, etc. all come from the Runner (JUnit, RSpec, PHPUnit, Test::WWW::Selenium, etc.) not Selenium itself. Once that is explained, it both expands the world of possibilities that opens up since people realize they get everything they had in the language before – as well as the ability to control browser, and also limits their expectations as they realize that Selenium (and automation) is not the Silver Bullet they might have thought it was.

In automation in general, one conversation I tend to have a lot is the whole metrics/ROI question which to me lies in the land of rainbows, lollipops and unicorns. The real value in automation, to me, is the ability to do more focused, high-value manual exploratory testing sooner in the process than if you didn't have the automation in place. How do you measure that?

Similarly, how do you measure the more complete model of the application you are testing? Writing sophisticated scripts means delving into the inner workings of the system to understand the data flows, to discover your true oracles, etc. One cannot help but learn more about the system. In my experience, the true value of automation (using the overly simplistic and often misleading metric of bugs found) is actually during the creation phase while you are digging around in this manner. Unless you have sloppy development practices with lots of regressions, then your automation should never find another bug once it is put into the main script pool.

REGISTER NOW! ◊ EARLY BIRD – FEBRUARY 10 ◊ SAVE $400

The Leading Conference On Software Testing

www.STPCon.com ⚜ 877.257.9531

Conference Tracks

Leadership Perspectives for Testers	Test Strategy and Design	Performance Testing	Test Automation	Agile Testing	Mobile Testing

Keynote Presentations

Tuesday, March 27, 8:00am – 9:15am

Testing Wanted: Dead or Alive

Jon Bach,
Director of Live Site Quality, eBay

A few speakers in the software test industry did a strange thing at testing conferences last year. They told their audiences "testing is dying." People like that seem to believe that when programmers write unit tests and conduct mandatory code reviews, it means no one needs to worry about bugs. They'll cite that because of "self-healing software" and the ability to hotfix a bug on the web, it means the role of the tester doesn't matter anymore.

Jon Bach disagrees. He's finding that despite the trends and advances in rapid deployment and automation, exploration and sapience is more crucial than ever. Given that there are so many variables and contexts in how the world operates, testing becomes more alive. There's value in having testers separate from programmers – staff who can get away from their keyboard now and then to see the big picture. Join us as Jon Bach explains why this industry needs people to see gaps between what customers experience and what automation is being run.

Tuesday, March 27, 1:00pm – 2:00pm

The Essence of Providing Feedback

Scott Barber, *CTO, PerfTestPlus*

Dawn Haynes,
Senior Trainer, PerfTestPlus

How effective is the feedback you provide to others? How valuable do you tend to find feedback that you receive? Do you know what your native critique style is? Join Dawn Haynes and Scott Barber for this interactive and amusing keynote session to find out.

Through a combination of role play and audience involvement, you will explore the attributes of your own style of critique, the relative effectiveness of common critique styles across several dimensions, and whether or not you have a knack for critiquing feedback. Bring your sense of humor, an open mind, and a willingness to participate in sharing a little well-natured, bi-directional feedback with your conference peers to this session, and leave with some new insights into giving and receiving feedback.

Wednesday, March 28, 8:00am – 9:00am

No Excuses! Incorporating Core Values, Accountability & Balance Into Your Career

Jay Rifenbary,
President and Author, Rifenbary Training & Development

Based on a solid foundation of core values, this presentation teaches, validates and supports the importance of key principles and skills such as, self-responsibility, organizational accountability, integrity, personal honesty, professionalism, self-respect, forgiveness, passion and positive attitude as they pertain to one's personal and professional success, and the success of others. Ultimately, this positively contributes to the morale and productivity of the individual and organization, resulting in a higher level of performance and a greater level of personal understanding, leading to a more productive, balanced and fulfilling career and life.

In this session, you will learn to: identify a foundation of core values applicable to the enhancement of both your personal and professional life; define those core values in regard to how they pertain to the overall mission and vision of the organization and develop an interactive blueprint of the relationship between the core values presented.

Wednesday, March 28, 1:15pm – 2:00pm

SoLoMo Is Shaking Up The Testing Landscape: How QA Organizations Can Adapt, Keep Up & Thrive

Matt Johnston,
CMO, uTest

The collision of social, local and mobile media (a.k.a. SoLoMo) is disrupting the testing industry and impacting software engineering and testing organizations across the globe. With so much sensitive and critical data flowing to and from SoLoMo technologies, there is immense pressure to ensure that apps are reliable, scalable, private and secure across a multitude of criteria, including handsets, tablets, OS, browsers, carriers, languages and locations. For instance, the adoption of social networking apps present numerous security hurdles. Companies that communicate with customers and partners via Facebook or Twitter must protect sensitive data. GPS apps present location testing challenges. Many apps are being used outside the sterile confines of the testing lab.

Using real-world success stories from Google, Microsoft, and others, Matt Johnston identifies how SoLoMo has transformed the software industry and reveals the secrets to overcoming the challenges that SoLoMo technologies present today.

Thursday, March 29, 8:00am – 9:15am

We Are the 99%

Anne-Marie Charrett,
Testing Coach and Trainer, Testing Times

Sometimes it feels that we have little control over what might happen in the future. It seems that wherever we look, there is doom and gloom about testing. But we do have control over some things. We can choose how we approach our testing, how we respond to change, how we apply ourselves to learning new things, how strong our testing community is.

In addition, in each of us there is a little bit of testing expertise. It might not be as brightly packaged and obvious as industry "experts," but it's there. Do you know what your testing brilliance is? By focusing ourselves on our strengths and our testing community we can become stronger, more adaptable and more confident about facing an uncertain future.

This keynote is a mix of stories and challenges that encourage us all to keep learning, to be open to change and to grow and build a healthy and vibrant testing community.

Register by February 10th Save $400.00

Track Descriptions

Leadership Perspectives for Testers

Understanding the business side of testing is as important as amplifying our approaches and techniques. In this track you will learn how to effectively lead diverse test teams, communicate with stakeholders, and advocate for testing. This track includes topics related to building test teams, leading business testers, handling the challenges of remote teams, communicating the value of testing, and using metrics in the decision making process.

Test Strategy and Design

Before you begin testing on a project, your team should have a formal or informal test strategy. There are key elements you need to consider when formulating your test strategy, if not, you may be wasting valuable time, money and resources. In this track you will learn the strategic and practical approaches to software testing and test case design, based on the underlying software development methodology. This track includes topics related to developing a test strategy, test estimation, risk management and data management.

Performance Testing

Performance Testing is about collecting data on how applications perform to assist the development team and the stakeholders make technical and business decisions related to performance risks. In this track you will learn practical skills, tools, and techniques for planning and executing effective performance tests. This track will include topics such as: performance testing virtualized systems, performance anti-patterns, how to quantify performance testing risk, all illustrated with practitioners' actual experiences doing performance testing.

Test Automation

Which tests can be automated? What tools and methodology can be used for automating functionality verification? Chances are these are some of the questions you are currently facing from your project manager. In this track you will learn how to implement an automation framework and how to organize test scripts for maintenance and reusability, as well as take away tips on how to make your automation framework more efficient. This track includes topics related to test automation architecture, continuous integration, agile automation, and what other tasks you need to perform after you designed your framework and scripted your test cases.

Agile Testing

More and more software development organizations are "going Agile". This track will help participants understand how they can fit traditional test practices in to an Agile environment as well as, explore real-world examples of testing projects and teams in varying degrees of Agile adoption. This track will include topics related to challenges in Agile testing, utilizing Agile testing techniques, coaching an Agile test team, and using games to internalize Agile principles and practices.

Mobile Testing

The rapid expansion of mobile device software is altering the way we exchange information and do business. These days smartphones have been integrated into a growing number of business processes. Developing and testing software for mobile devices presents its own set of challenges. In this track participants will learn mobile testing techniques from real-world experiences as presented by a selection of industry experts. Topics in this track will include techniques for mobile testing, the latest trends affecting mobile testing and a framework to manage mobile testing.

Breakout Sessions

TUESDAY ◊ MARCH 27

9:30am – 10:45am

101 Conscientious Testing: Understanding the Ethics of Being a Tester

Mark Tomlinson, *Shunra Software*

This session will discuss the ethical decisions that testers face every day and will provide guidance on how to formulate proper reporting for unethical or illegal issues.

102 Test Strategies from Around the World

Rex Black, *RBCS*

This presentation discusses the significant test strategies in use by testers around the world and how to blend these strategies to achieve optimal results.

103 Threads: The Good, The Bad and The Ugly

Claire Cates, *SAS Institute, Inc.*

The pros and cons of threading applications and the performance problems that can be discovered will be discussed in this session.

104 Lightsabers, Time Machines, and Other Automation Heuristics

Adam Goucher, *Element 34*

Discover how heuristics can help identify why automation rollouts are unfolding in a certain way and can point to possible pitfalls along the road.

105 An Outlook to Mobile Testing

Rumesh Palaniswamy, *Cognizant Technology Solutions*

Recognize the critical factors of a successful mobile testing program and the strategies and approaches for efficient and error free delivery.

11:00am – 12:15pm

201 Coaching Testers

Anne-Marie Charrett, *Testing Times*

Coaching is an effective way of increasing skill in a real and tangible way. Attend this session to learn the how's, the why's and the benefits of coaching.

202 Strategies for a Successful Systems Integration Test

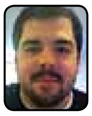
Fiona Charles, *Quality Intelligence, Inc.*

This session will present key strategies to help you address the important questions, and craft a manageable and effective systems integration test.

203 Planning For The Perfect Storm: The Value Of Peak Testing

Scott Moore, *Northway Solutions Group*

Learn tried and true guidelines that can be used to plan for realistic peak loads and understand what worse case looks like before customers do.



204 Forming Your Own Selenium/Watir Clouds for Fun or Money

Aaron Broad, *PQA, Inc.*

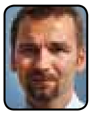
Attend this practical session to learn how to form your own Selenium/Watir cloud which is much easier than you think!

205 Agile Testing: Facing the Challenges Beyond the Easy Contexts

Bob Galen, *iContact*

Learn to balance the requirements dictated by challenging contexts with those embraced by truly agile testing parameters.

2:15pm – 3:30pm



301 The Right Stuff: Origins of A Test Manager

Drazen Tomac, *Nagravision – Kudelski Group*

Learn what characteristics make up a good test manager given that agreement can be reached that such a person exists.

302 Creating a Security Test Plan

Dan Alloun, *Intel*

Learn the steps and guidelines for creating a security test plan as well as a method to objectively assess the risk of vulnerabilities found during execution phases.

303 Lessons Learned in Performance Testing – Cloud Edition

Ben Simo, *GoDaddy*

This session will focus on real-world examples of developing and testing highly-redundant distributed cloud storage services.

304 Data Warehouse Test Automation For Sapient Testers

Eric Jacobson, *Turner Broadcasting Systems, Inc.*

Based on personal experience, this session will suggest a test approach that allows sapient testers to write automated data warehouse tests.

305 Mobile Ecosystems and Their Impact on Testing

Uday Thonangi, *Progressive Insurance*

Test management challenges dictated by continual release of new devices, browsers and operating systems are explored in this session.

3:45pm - 5:00pm

401 Developing a Professional Testing Culture

Greg McNelly, *Progressive Insurance*

The path leading to the professional testing culture at Progressive will be shared as an inspiration for changes at your organization.

402 Why Tests Don't Pass or Fail

Doug Hoffman, *Software Quality Methods*

This sessions explains why tests don't necessarily pass or fail and includes some suggestions for ways to benefit from this perspective about test outcomes.

403 Performance Testing Metrics and Measures

Mark Tomlinson, *Shunra Software*

Gain an expansive understanding of performance testing metrics and how they can be used to drive quality initiatives in the application lifecycle.

404 Automated Testing Strategies for Databases

Stephen Vance, *uTest, Inc.*

Learn to address challenges that come up when automating testing of data-intensive applications, including test repeatability, performance, and transactional integrity.

405 Fitting In: A Case Study in Agile Team Structures

Catherine Powell, *Abakas*

Understand the dynamics of an agile environment and how to effectively manage the test team within it.

Register by February 10th ⚜ Save $400.00

Breakout Sessions

WEDNESDAY ◊ MARCH 28

9:15am – 10:30am

501

Breaking the Chains: Envisioning Tomorrow's Test Management

Lynn McKee, Quality Perspectives

Discover the new perspective that must be embraced to enable and empower highly successful testing teams.

502

A Method of Assessing Testing Risk Mitigations

Geoff Horne, *ISQA*

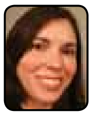
Learn a proven method for identifying and appropriately assessing risks and their mitigations for testing projects.

503

A to Z Testing in Production: TiP Methodologies, Techniques, and Examples

Seth Eliot, *Microsoft*

An introduction to TiP including methodologies attendees can use illustrated with examples from Microsoft, Netflix, Amazon and others.



504

Where'd My GUI Go? Testing APIs

Catherine Powell, *Abakas*

This session is a survey of testing various types of APIs with non-GUI tools and includes some sample automation using Python and Ruby.

505

Test Virtualization: Revolutionizing Quality in Organizations

Jamey Jones, *Dylabs, Inc.*

Learn how implementing virtual test labs can build strong relationships among multiple groups resulting in tangible benefits across the entire organization.

10:45am – 12:00pm

601

The Future of Testing: Managing the Inverting Testing Pyramid

Harish Narayan, *Vistaprint*

Explore the organizational, business and technical challenges facing testers of the future and learn innovative levers to overcome these challenges.

602

Surviving an FDA Audit: Heuristics for Exploratory Testing

Griffin Jones, *Congruent Compliance*

This presentation explains a model to prepare for an audit or to baseline your current practices for an improvement program.

603

Web Load Testing for Dummies

Scott Barber, *PerfTestPlus, Inc.*

Understand the differences among performance testing inside your network, outside the firewall, and through actual end-user devices/ locations.

604

Cutting the Mustard: Lessons Learned in Striving to Become a Superstar Agile Tester

Nancy Kelln, *Unimagined Testing Inc.*

People are the key to a successful Agile project – learn to challenge yourself and become a superstar on a successful Agile team!

605

What Mobile Developers Wish Mobile Testers Knew

Wilson Mar, *HP*

Examine how testers and developers can manage risks together to stay ahead of the complexity and ambiguity in development of mobile apps.

2:15pm – 3:30pm

701

Talking to Triangles

Dawn Haynes, *Consultant and Sr. Trainer, PerfTestPlus, Inc.*

Learn to communicate key issues and interpret vital information from the team as it relates to the needs of the project.

702

Managing COTS Test Efforts: It's Like Conducting an Orchestra!

Virginia Reynolds, *Insight Quality*

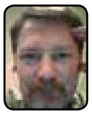
Walk through a high-level testing strategy to test any type of enterprise COTS system – patch, upgrade, or new deployment – based on lessons learned the hard way!

703

Troubleshooting Application Performance Issues

Kerry Field, *US Bank*

This session will present examples of real life application performance issues to show how successful resolution depends on effective people, process and technology.



704

Automation Isn't All Shiny Toys

Jim Holmes, *Telerik*

Join this discussion highlighting the value automation brings to every software project while learning proven ways to avoid common automation pitfalls.

705

The Hard Stuff: Questions About Agile

Matthew Heuser, *Excelon Development*

Bring your most difficult questions about Agile to this interactive panel discussion featuring experts from the industry!

4:30pm – 5:45pm

801

A Testers Guide to Identifying, Managing, & Eliminating Technical Debt

V. Lee Henson, *AgileDad*

Learn what technical debt is, and steps you can take to eliminate it!

802

Adopting the Franchise Model in Testing Software

Kat Baum, *Ameritas Life Insurance Company*

In this session the franchise concept is explained including how it works, how to use it in testing, and its benefits and challenges.

803

What Happens When Agile Testing Falls Behind?

Robert Walsh, *Excalibur Solutions, Inc.*

Discover proactive and reactive testing strategies for keeping pace with development in an agile environment.

804

Web Consistency Testing

Kevin Menard, *Mogotest*

Learn about an automated issue detection system that makes Web Consistency Testing available to every person in an organization, from product manager to QA engineer.

805

Don't Go Mobile Testing Mobile – Real Life Experience Testing with Mobile Devices

Brett Masek, *Lash Group*

Gain valuable insights on implementing a mobile testing strategy from this session based on real life lessons learned!

Breakout Sessions

THURSDAY ◊ MARCH 29

9:30am – 10:45am

901

Rightsizing Testing

Scott Barber, *PerfTestPlus, Inc.*

Attend this session and learn a problem solving approach and some valuable tips for right-sizing your software testing efforts.

902

Evaluating and Improving Software Usability

Philip Lew, *XBOSoft*

Attend this session to understand the relationships between usability and user experience and to explore measurement and evaluation methods for improvement.

903

Performance Testing within a Performance Engineering Center of Excellence

Brian Lynch, *BNY Mellon*

This case study will familiarize performance test teams on how, why and where performance testing fits within a PE CoE at BNY Mellon.

904

Test Management in your Continuous Integration Process with Jenkins and TestLink

Bruno Kinoshita, *TupiLabs*

Learn to spot how testing is present in a continuous integration process and what is necessary to manage your tests in an efficient and integrated manner.

905

Crowdsourcing Testing – Fostering Innovation through Collective Intelligence

Ralph Decker, *AppLabs*

Understand what is required in order for Crowdsourcing Testing to be rewarding, innovative and successful.

11:00am – 12:15pm

1001

Software Testing at Startups

Michael Kelly, *DeveloperTown*

Recognize your current product testing needs and learn how to adapt and scale to meet the needs as they change during a period of growth.

1002

Testing Small Projects: Strategies for Successful Delivery in 2 to 12 Weeks

Lanette Creamer, *Spark Quality LLC*

Success on a small project depends on your ability to adapt so that the goals you are meeting directly serve the stakeholder and business goals.

1003

The Test Car Driver

Mike Bonar, *Bonar Consulting*

Learn who it is, exactly, that we test for and why, as testers, we do what we do drawing on historical parallels from the automobile industry.

1004

Extreme Mobile Testing with Selenium (and Robots)

Jason Huggins, *Sauce Labs, Inc.*

This talk will show developers how to use Selenium to test mobile applications in 3 different ways including using a robot known as the BitbeamBot mobile testing platform.

1005

Looking at Code from a DevTester's Perspective

Stephen Vance, *uTest, Inc.*

Gain insight on how to critically look at code to improve its testability and verify its intent as expressed through its structure and syntax.

www.STPCon.com REGISTER now 877.257.9531

by DevyaniBORADE

Managing Test Data

The challenges of testing on live websites

Angelina Jolie is my customer. Tom Cruise shops at my online store. Sachin Tendulkar has often worked for me collecting meter readings. Wayne Rooney is a frequent supplier who ships my products to their destinations, taking care of fulfilment and logistics. Bill Gates features regularly as a job-hunting candidate when I am posting online jobs. Even Albert Einstein has managed some general administrative work behind the desk for me.

Before you think I am talking through my hat, allow me to tell you that these are all only "user records" in my software application testing!

For those of you who are privileged enough to have massive automating applications that generate gigabytes of test data at just the click of a button, you will probably have a fair idea of how high a proportion of the total time and effort of testing actually goes into simply getting the test environment set up and the test data created. For those of us who aren't so lucky, we pretty darn well *know* it.

I work with a software solutions provider that specializes in creating bespoke web applications for businesses. These web applications usually have some form of ecommerce or other. When we release a version of the application, it is first tested on a test or "staging" site and once approved by the relevant stakeholders, it is deployed to "live". However, as with any software, there is always a guarantee that something will go wrong. Therefore, in addition our standard release management, configuration control processes and deployment checks that are in place, we believe that it is best and safest to give the "live" site a quick once-over to ensure that everything works as it should. This involves typical tasks like performing searches, page navigation, registering as a new customer, logging into the customer account, putting through a purchase and checking the order details are all captured correctly. Inevitably, this creates test data in the client's live production environment – data that is of no use to anyone once the test is successfully performed, *data that is just sitting there and cluttering up the system.*

Can this be avoided? No. Connecting to a test database which is a copy of the live one, even if it is only for a brief moment or two, is not an option. The site is live and may be experiencing genuine traffic at any point of time, which cannot be diverted to a test database.

So can this be fixed? We could run scripts at regular intervals to delete test data. But most developers, however seasoned veterans they might be in the software development arena, hesitate to delete anything on a live working system for obvious reasons.

And yet there is a very real need to remove such data. System administrators have access to user and transaction records and can see such test data. Worse, some types of test data, for example new products and product reviews, can be seen on the website itself. Potential customers could get turned off by an unprofessional looking "asdfqwsdgj llpouoiuylkmnb" staring at them from under the welcome message, or resent what they may think are fake reviews intentionally designed to boost the website's profile, thus losing our clients valuable business. This is clearly unacceptable.

The challenges don't stop there. Test data, if left untouched and unhandled, may interfere with the client's business intelligence. For an application that goes through one or two new versions a year, this aspect may not be threatening. However, the interference may be significant for a more dynamic site that undergoes frequent maintenance and upgrades and as a result has more test data on it. Say a manager generates a monthly report of how many new users have registered on their site in the last month or which products are the most popular buys. If, out of 50 records returned, 40 are test records, the manager isn't going to be very pleased with the outcome!

When I first started testing software, I would create the usual TESTING and TEST123 type of records which could be easily identified as test data. After a while, I began to run out of combinations with the word TEST in them. Besides, it all began to get a little tedious and prone to mistakes. Did I just create an invoice with reference TESTED or was it TESTER? Was my address TEST_123 or simply TEST123? Oh dear.

Then I hit upon a simple solution: I populated fields with their captions and started suffixing numbers to them. Brilliant. Now I could tell by looking at HOMETELEPHONE1 that it was the value in the Home Telephone field and if it started appearing in the Mobile Telephone field, there was a bug. Again, for a while, I thought I had it all mapped out. Surely with this precise strategy there was no chance of making mistakes.

But is there ever such a thing as a perfect system, even one so essentially uncomplicated as a naming system? Of course not. Pretty soon I woke up to the fact that I had about ten FIRSTNAME1's and about seven LASTNAME99's in my database because each day I created a few records, and each subsequent day, having lost count of where I had last reached the previous day, I ended up creating them all over again.

Right. Next strategy. I considered using my own name everywhere. But how would I differentiate between one record and the next?

So it was back to the drawing board. I thought long and hard. What was the one thing that was unique and seemed to be available in abundance in the world? It didn't take me long to realise the answer – celebrities! You couldn't escape them. Even if you retreated into the interiors of a third world country, sooner or later a Madonna or a Bono would make an appearance and burst into your consciousness like air from a flat tire, forcing you to acknowledge their existence.

Putting my plan into action was for me the work of a moment. The beauty of using celebrities names as test records was that *a.* they were real names and made sense, *b.* they were easily recognizable as being test data (indeed, no matter how much I would like Beyonce Knowles to actually purchase some boat anchors via my website, it's hardly likely to happen) and *c.* by virtue of resembling real world data, they sometimes threw up interesting behaviour in the applications (if only I had a dollar for each time Mary Flannery O'Connor had caused the module to crash!).

So Agatha Christie became my student, Winston Churchill searched for a job as a candidate, while Richard Branson happily wore a grass skirt and wielded a mean sword in a gaming application. The office often rang out in merry laughter when I made statements like, "Hey, where's David Beckham's order gone?" or "Oh no, I can't see Lady Gaga in this list!" aloud.

An interesting issue spawned at this point – would using a name like ADOLF HITLER have any fallout? Would users be offended if they happened to come across such test data? After all, the name itself is a perfectly valid combination of characters. Perhaps some people would rather see a meaningful name like "Adolf Hitler" than a nonsensical name like "asdfgdsfh yupoeuty" which is what most people bang out on the keyboard whenever it comes to data entry. Assuming a computer program had generated this name after randomly selecting from a list of First Names and Last Names, would our clients throw the computer program out, where an industry professional would only see simple string values and read no more into it? Can one argue that there isn't only one Adolf Hitler in the world and that using this particular name merely as test data is not meant to insinuate, taunt, justify, remind, disrespect or condone the real person or real actions? Would this sentiment change if one were to use a name like MOHANDAS GANDHI for a user record in a computer program for, say, the alcohol industry? (The man Mahatma Gandhi was a teetotaller and passionate in his beliefs against alcohol. In his honour, on the anniversary of his birth every year, Indians across the entire country observe a 'Dry Day' and no alcohol is legally sold or purchased anywhere.) Would a name like GEORGE MICHAEL appall homophobes? Conversely, would a tester be applauded if he/she were to use a name like MOTHER TERESA or NELSON MANDELA as test data?

Over time, I proposed several other solutions to the problem of dealing with test data in live production environments:

We could create a new flag on every single table in the database and run the application in test mode by setting this flag. All reports generated would subsequently ignore test data that was thus flagged. This was greeted with several vociferous 'nay's by the developers because it involved considerable rewrite of each of the applications and they would have to bear the brunt of the work. Also, running an application in test mode would mimic the live environment, sure, but it would not *be* the live environment itself, which sort of defeated the purpose.

We could offset the test data with more test data. In other words, for every test user who wrote a glowing review for a product, create one who completely dismisses it. Again, this was rejected as not being universally applicable to every situation.

We could create some business rules that would ignore test data beyond a certain threshold or coming from a certain username or login or role. This, unfortunately, rather limited the test cases to a very small number and forced the testing to become more of a regression test than an end-to-end high-to-low elastic sanity test.

We could also ask some candidate clients to beta-test the application themselves with real users, but they would not have the know-how, skills or expertise to do this competently and there was the risk of this reducing their confidence and faith in our services and customer commitment.

In the end, I recognized that the way to create test data for a live environment would have to be determined after taking into consideration the specific end user concerns, functionalities and reports specific to the particular application under test. For our applications, we agreed to create a defined list of test data of all possible types – user names, products, general textual matter et al – and decided to use only this pre-defined list when testing on live websites. We published this list across our organization and also informed our clients about the exercise. System administrators were advised to ignore such data when processing reports or performing other administrative duties like validations or email interactions. At the moment, we are piloting this scheme and so far it seems to have been going smoothly with no hiccups. However, as our clients' businesses grow, we realize that there will probably come a stage at which this system will cease to function effectively and efficiently. At that point, we will need to start looking for a new strategy. Hopefully, by then, our applications will have matured and built testability into themselves. But that is a topic for another day.

About The Author

__Devyani Borade__ is a software tester with a web solutions provider near London. In the past she has been a programmer, a business analyst, a team leader and a quality manager. Her unofficial title is "Mighty Tester" and her motto is "The test is mightier than the code". Contact her via her website Verbolatry at http://devyaniborade.blogspot.com.

Join the STP Community

Visit us Online. Create or Update Your Profile.

SOFTWARE TEST PROFESSIONALS
Community
Events
Training
Resources
ST&QA Magazine
Community Forum
Search
The Latest News
TEACHERS & ROCKSTARS
Closing the Software Testing and Development GAP
Deliver Value with TESTING METRICS
MOVE BEYOND THE MYTH
STP ONLINE SUMMIT DECEMBER 13-15, 2011
Current Issue
STQA
5 Ingredient Quality Metrics Vowel Soup
Web App Architecture
Can Your Website Handle the Traffic?
Latest Resources
Articles

Basic Membership:

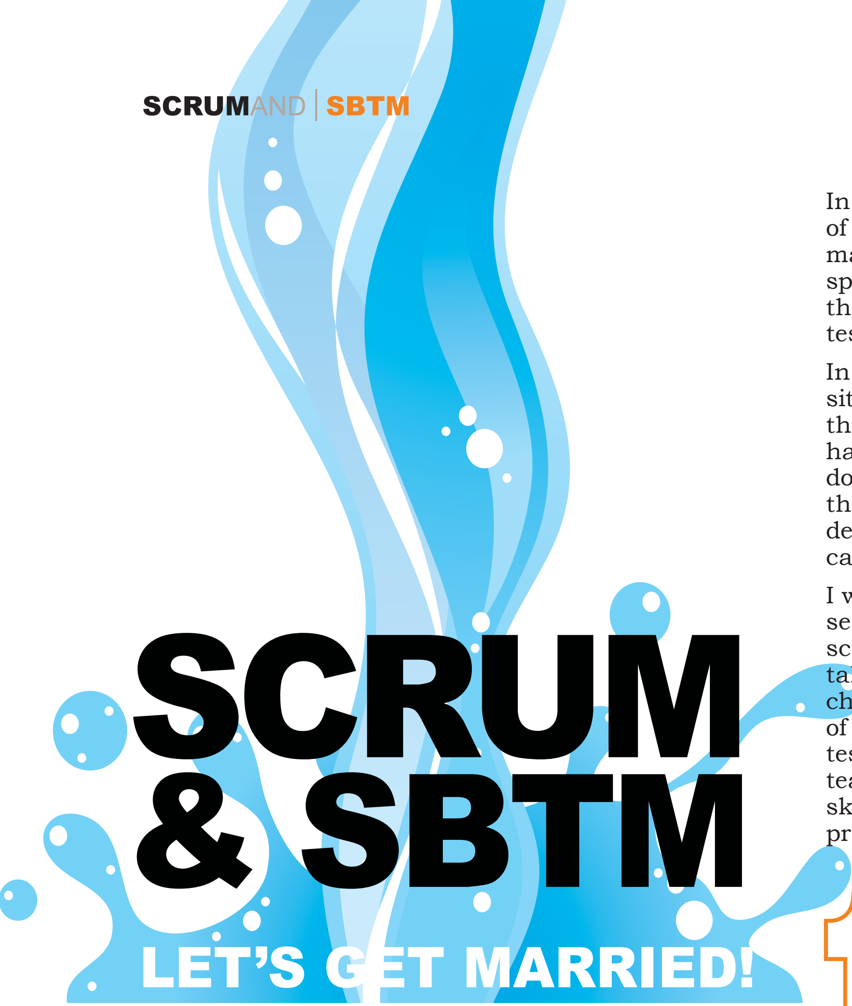
- It's Free
- Your thought-leadership will be considered for the ST&QA magazine, TQA and Insider e-newsletters
- Access to articles and research
- Receive invitations to webinars and podcasts
- Access to the archived e-newsletters and magazines
- Receive invitations to conferences and training events (both online and in-person)
- Earn badges and gain visibility by your peers
- Ability to fully utilize the website features including rating and commenting on articles and joining a Crew
- Receive the ST&QA magazine and e-newsletters
- Free Focus Book

... Did we mention it's FREE!

www.softwaretestpro.com
is your one stop resource for educational, career and networking resources.

Join Today!

STP SOFTWARE TEST PROFESSIONALS



SCRUM & SBTM

LET'S GET MARRIED!

I often see scrum teams dealing with testing in two different ways, either they kick testers out or the testers get pushed into the team without knowing what they should do and then try to do waterfall testing in scrum.

by Henrik **ANDERSON**

Neither of these alternatives are particularly favorable. In the first scenario having heard about all the cool ways you can automate testing, they talk about TDD, BDD, ATDD, CI tests, cucumber, fitness, specflow and others. What they do not realize is that when they think they are testing, they're merely *checking*; that is predefined checks that confirm our existing beliefs about the software. There is no *testing* going on; that is finding new information about the product by exploration, investigation, discovery and learning.

In the second scenario, testers hold on to their old ways of testing, waiting for complete requirements. Writing manually scripted tests is really hard to fit in to a sprint and tends to result in testing being squeezed into the end. You never get done with what you intended to test before the sprint ends.

In this article I focus on what you can do in this latter situation when you have (and I strongly recommend that you have) testers on the scrum team. If you do not have testers on the team how do you know that you are done and have a potentially shippable story? I assume that you have needed checking in place and that developers take responsibility for this so we testers can focus on providing value by testing.

I will go through a sprint and write about how I get session based test management (SBTM) to fit into scrum. This is not a complete guide and I will not talk about the basics of either scrum or SBTM. I have chosen to highlight five points that I find critical and of particular interest. I hope this can help you as a tester to be productive and highly valuable in a scrum team. However, this still assumes that you are a highly skilled tester; there is no way around that. I'm just providing you with a framework.

1 Sprint Planning –

Where you decide what to do for the upcoming sprint.

The whole scrum team gathers together with the product manager, who explains the user stories that have the highest priority and we ask questions to get a better understanding of the story. Remember, a user story's prime purpose is to encourage discussion to get a common understanding of what is needed to be done, not specify every requirement in detail.

When developers start splitting the story into tasks, we testers go to the other end of the room (assuming you have a large room) to plan for testing. We do this by first calculating what I call "testers velocity". It is the total number of sessions possible over the sprint minus planned out of office time minus planned other things (meetings etc) equals actual available number of sessions for the sprint.

Example: we run three sessions per day and we are two testers and the sprint is 15 working days = 90 sessions. However, one tester has a one day training scheduled and the other one has a half day doctor's appointment. In this scenario, three plus two equals five sessions. We also have an all employee meeting which removes one session for each tester thereby leaving two sessions. That gives us 90 minus 5 minus 2 which equals 83 available sessions for the sprint.

We split the available sessions over the stories (don't forget to include non-story related work) we intend to test over the next sprint. Usually we do not distribute them evenly; we consider risk, complexity, size, value, experience and other things. When we have the distribution we come up with missions on what to test for each story and we estimate how many sessions we need to fulfill the mission. When this is done we do a final check that we believe the distribution is doable and that we believe we will reach enough testing. At the end we present our plan for the rest of the team and if needed adjust it based on feedback. Now keep in mind that we have just made a plan that provides direction, this is not written in stone.

2 During the Sprint – Now things are happening!

A good scrum team delivers completed stories at least every second day during the sprint (if your team does not do this, you have to work harder to slice your stories smaller). This means that we testers do not sit and wait before we get things to test and we can rapidly provide feedback to our team on the product we are building. Each week we look at which stories we aim to test and in consideration of the mission(s) we have for them we produce charters and schedule them over the week. Again this is not written in stone. Since we continually learn and get new information when testing, the weekly plan we have on Monday never looks the same on Friday; we have changed charters around and exchanged charters for more important ones.

3 Daily Stand Up Meeting – Reporting to our team on what we are doing.

Every morning the scrum team meets for a short status meeting. Normally you answer three questions (What did you do yesterday? What will you do today? Do you have any obstacles?). We testers report the overall health of the product and any specific status on tested stories. We usually use adaptations and variations of James Bach and Michael Bolton's "low tech dashboards" to visualize test progress and status. We split the product into different areas and report on what we plan to test and what we are testing and the health on that particular area. We also use a burn down chart of the test sessions for the sprint. I do not write much about this since there are so many ways of doing this and the most important thing is that your reporting supports your purpose.

4 Bugs – How do we deal with them?

First, not all bugs have to be fixed. However, all bugs have to be considered and the question regarding for whom this bug is a problem should be asked. We put the bugs up on the scrum board as soon as we find them. Log files and other information related to a bug are just stored on a server – we keep these things simple and rapid. We use a different color on the bugs so that everyone easily gets an idea of the current health. Sometimes we set a principle that no new work can be started unless all bugs have been considered. This means that if there is a bug on the board that no one has looked at they have to deal with that one before starting a new story. This does not mean we have to fix it, it can equally well be that we (together with the product owner) decide not to do anything with it. Then we hand the bug over to the product owner who can either throw it away or put it in the product back log for future implementation. Then it is no longer a focus for the team and also not visible on the scrum board any longer. In reality we find that since we look at the bugs as soon as we have the information about them and we see that it is a smaller thing to fix therefore we tend to do it immediately. In that way we keep our code base continuously stable and avoid degradation over time. The important thing here is that we deal with the bugs immediately while the information is fresh.

5 The End of the Sprint – Close down.

When the sprint ends we take a snapshot of our test dashboard and store this on a server. We also have all executed charters with the session notes stored. This is the documentation that we found usable. We do not take special consideration to the bugs since they either are fixed or handed over to the product owner. We have not found it useful to store them for any historical analysis but others may have that need. We have a deeper discussion with the product owner on our findings during testing and discuss if there is something the product owner should consider regarding the health of the product. Since we continuously over the sprint kept the feedback loops short and tight we do not have the need for a big reporting activity at the end. The team has been informed during the sprint. After a demo and a retrospective we are ready to take on the next sprint.

I hope you find these points useful. This is a collection of things that I've seen different teams and companies struggle with. My intention is not that you should copy this without questioning but use the suggestions if it is a good fit for your team. I hope that it will give you some good ideas of how things can be done.

About The Author

Henrik Andersson *is founder of House of Test, a context driven testing consultancy and outsourcing based in Sweden and China. Henrik operates mainly in the Nordic countries by helping companies increase efficiency and reconstructing their testing to deliver better, faster, cheaper and with improved value. He provide leadership and management consulting in the fields of Testing and Agile development. He tests, coaches, consults, speaks, writes, manages and thinks about software testing and problem solving.*

Henrik is well recognized in the Context-Driven Testing community.

"But in the larger picture, they weren't bad interruptions, because they did provide value to the end product. And isn't that our real goal as testers; to provide value by finding technical quality related information about the product."

– BernieBERGER

A DAY in the Life of a **Software Tester** PART 3

by BernieBERGER

AM 7:45

"ATTENTION METRO TRANSIT PASSENGERS. THERE IS A DISABLED TRAIN DIRECTLY AHEAD OF US. WE WILL BE MOVING SHORTLY. WE THANK YOU FOR YOUR PATIENCE AND APOLOGIZE FOR THE SERVICE INTERRUPTION."

"Interruptions", I mutter to myself on the crowded train this wet, rainy morning. How I really hate interruptions. They always come at the worst moments.

I'm still half asleep, standing, holding on to the overhead horizontal bar, and my hooded rain jacket dripping onto the floor.

"Interruptions." I say to myself, again. The tester in me can't help but to analyze the concept. At this time of day it's more like a stream of consciousness:

"What's an interruption, anyway?" Well if you assume that there is a plan or some expectation and then a foreign element comes along and blocks the progress of that expectation for a while, that's all it is –I think. There are probably different kinds of them, too. There are the simple kinds, like when someone is talking and someone else jumps in with something to say before the first person is finished speaking. Man, it really bugs me when that happens. Another example is when you're watching a television show, and a commercial comes on, that's kind of an interruption too. Here on this train, even if it wasn't stuck right now, when it makes its stops picking up other passengers, maybe that's also a kind of interruption to everyone who is already on the train.

We've had quite a few interruptions at work lately. Like on Monday, when we were going to test this new build for compliance, first our plan was interrupted with a bad build. Then we had that database corruption. Yesterday's doozie was when our test messages were being sent to production. Now the project manager thinks we're behind schedule because of these.

But were these unplanned events really interruptions? And if so, what kind were they? Well, they *were* external elements that blocked the attainment of a goal. But in the larger picture, they weren't bad interruptions, because they did provide value to the end product. And isn't that our real goal as testers; to provide value by finding technical quality related information about the product. In a way, and certainly from the perspective of the project manager, these were interruptions. But on the other hand, they were productive interruptions.

Well, in any event, I just hope we can finish up this project today. It hasn't exactly gone according to plan, so far but maybe that's just the point. These things that came up over the past two days are not so much interruptions, but the rational adaptations of the initial plan to expected but unknown changes. Maybe there are good and bad interruptions.

The train starts to move again. And so the wet morning commute continues.

About The Author

__Bernie Berger__ (bernie@testassured.com) has been testing software at Wall Street firms for more than 15 years and has experience on various software projects related to the financial industry, including electronic trading and algorithmic strategies, FIX protocol and market data validation. He is a former director of the Association for Software Testing (AST) and is chairman of the AST Financial Services SIG. He founded and runs the Software Testing in Financial Services (STiFS) workshops.

8:30

Shaking off my dripping umbrella, I walk into the office and sit at my desk. Unlike yesterday, I'm one of the first people in the office. I browse emails as I sip my morning coffee. Not that many messages in my inbox this morning. That's good. Sorting emails into categories: things that need my attention, things that I need to save for future reference, and things that should be deleted. I try to map out my schedule for the day.

9:10

Raj comes in, soaking. "Man, I hate the rain."

"What happened to you? No umbrella?"

"No, I had one but the wind blew it away. I became this wet just from walking the last block to the office."

"Wow. Well pull yourself together. When you're ready let's figure out what we need to do today. I saw an email that they want to release this MMY thing today, if we can."

"Oh, really? OK, let's talk. Give me a few minutes, though."

"Sure. Take your time." I reply. "I'll be in the test lab."

9:25

Dry and a bit more tidy, Raj joins me in the lab.

"Alright, let's plan out the day. We still have a few more tests to run from yesterday. And then we'll need to test the new permissioning feature that Vlad was supposed to write. Do you know if he's done?"

"No, I haven't seen him yet."

"OK, so anyway let's get cracking on these tests. And we'll check with Vlad when he comes in."

11:00

Things are going pretty smoothly so far. We've been in a pretty good groove for the last hour or so, running tests on those areas left over from yesterday. We're ready to move on to the next area to test, which we decide should be the newly developed permissioning feature.

"Hey let's go pay our friend Vlad a visit." I say.

So Raj and I start walking to his cube. On the way, we see Bob, the compliance officer, talking with the project manager in the hallway.

"How's the testing coming along?" he asks.

"Pretty good, actually." I say. "We found a few issues over the past two days, and now we're ready to test the permissioning, which was just added. We're on the way to check in with Vlad."

"OK that's good to hear." says the project manager. He seems to be in a better mood than usual. "We're going to want to release this into production tonight, if we can. Just to get it out of the way. Otherwise, we'll have to release it later in the week along with all the other changes that is going in. It would be safer if we did it tonight, on its own. Just in case we need to roll back, it would be an easier process. Because if we need to roll back later in the week, it will be a mess."

"Yeah, that sounds reasonable." I reply. "We'll let you know how the permissioning tests go."

"That would be great", says Bob. "Because as soon as it's in production, we're going to have all our customers clamoring over this feature. They have been waiting for this for a long time, and now that they have it, they are all going to be submitting their MMY messages."

"Waitaminute..." I say, slowly. "Do you mean to say that all our customers will be submitting MMY messages all at the same time?" What's the expected system load on that?" I look at Bob, then the project manager.

"I don't know. They are all just going to be using it", says Bob.

"Concurrently?"

"Yes, concurrently", he answers.

The project manager chimes in, "I would hope that you guys have been doing capacity testing all along."

"Well, we did some, but not to the extent that you're talking about now. Frankly, this is the first I'm hearing about capacity testing being necessary for this. We did run our testing strategy by you when we started." I say, afraid of where this conversation may be heading.

Raj jumps in. "It's not a problem. I have a script that we can use. The one from yesterday, we used it to send one message every five seconds. I can easily modify that to increase load. And we can run it on multiple instances. That should be enough to test capacity. And we can monitor the system performance while they are running. "

"OK, let's do that, then. It won't be an elegant test, but it's the best we can do. And it will probably be good enough. Let's divide forces," I say to Raj, grateful that the hallway conversation just got reframed. "You focus on the load testing, and I'll go follow up with Vlad on the permissions."

"No problem."

"So let's take a checkpoint later again today to see where we are. Four o'clock?" asks the project manager.

"Agreed." we all say.

11:25

Raj goes back to the lab to work on the load script, and I finally, get to Vlad.

"Hey Vlad, how's it going?"

"Could be better, could be worse." he replies in his usual fashion. "What's up?"

"I wanted to know where we are with the permissions for MMY entries that we talked about yesterday. Is it ready for testing?"

"Of course it's ready, didn't you get the email I sent you last night? I was here till 11PM working on that."

"Email? No, I don't think I saw an email from you. But let me go check again, maybe I missed it."

I go back to my desk. This is not the first time that we've had problems with the email system. Sometimes the email server gets so backed up that it takes forever for messages to come through. Other times they never seem to make it at all.

Nope. No email from Vlad anywhere. I wonder if there were any other messages that I didn't get.

11:45

Phone rings.

"Hi it's Diane. I just wanted to let you know that the QA Center of Excellence meeting is going to be in room 4C instead of 6A today. I'm calling because some people were having problems with their emails. We're all excited to hear your presentation! Are you ready?"

I feel my heart drop into my stomach. The QA Center of Excellence meeting! I forgot all about that!

Diane is a QA manager in a totally unrelated area in the company from where I work. She runs a weekly lunch meeting for QA departments from different parts of the company, where someone usually gives a presentation about testing tips and techniques that they've used. Some of them are really interesting, but sometimes they are excruciatingly boring. I usually like to go when I can. Last week they asked me to talk this time. And I would love to do it, but I just don't think it's going to work out. I just don't have time for it now.

"Hey Diane, listen...I know this is really last minute, but I'm kind of in the middle of an emergency right now. We're in the middle of testing a release that we're trying to get out tonight..."

"Oh, don't tell me that you're going to cancel on me!"

"Yeah, I'm sorry."

"That's really too bad. We were looking forward to it."

"I'm really very sorry. I would have liked to do it, but I just don't think it's the best use of my time right now. Can we reschedule?"

"Yes, I suppose we could. We already have someone lined up for the next two weeks. How about the week after that?"

"OK, that should work. Again, I'm really sorry about it."

"Hey, it's OK. I'll send out the cancellation notice now."

"Alright. Thanks for understanding."

"No worries"

"OK, thanks."

"Bye. "

12:30

So the permissions feature is in. I start setting up the tests for that, when a crowd converges around my desk. Eric, Vlad, Kathy, and a few others.

"Are you coming with us out to lunch? Harry is here from headquarters, we're all going out to lunch with him. You coming?"

Oh, man. Harry is the senior vice president of technology for our division. Whenever he comes to town he takes us out for drinks after work, but sometimes he does lunch with whomever is free. He's a very friendly, outgoing guy, and it is good to be on his good side. I always try to go out with him when I can.

"He's here today? For how long?"

"One day only, he's going back tonight. Are you in or out?"

Decisions, decisions. I just blew off Diane and the Center of Excellence. But I really want to go with these guys. I do a quick reality check. Building a relationship with Harry is important to me, from a career development perspective. And he's going back to HQ tonight, who knows when he'll be back. The COE is different because nothing was lost, it was only rescheduled. And besides, I have to eat anyway. Suddenly I remember the book "Never Eat Alone", about personal networking, how to connect with another person's circle of friends. OK, I'll make an appearance, I decide. Thirty minutes away isn't going to be the deciding factor of the release tonight, anyway.

"Is it still raining outside?", as I grab my jacket.

1:30

Thirty minutes became an hour. I'm not really sure how that happened. Things just seem to take longer when it's raining, somehow.

But I'm back at my desk. Apparently the email server was rebooted because there are about 200 unread messages in my inbox now.

A meeting alert pops up on my screen.

Interview consultant candidate in 15 minutes.

"What the heck!? I have to interview someone?" Where did this come from? Besides, I thought we were still in a hiring freeze. What's this all about?

The phone rings.

"Hi its Clarie from HR, I just wanted to confirm that you'll be interviewing because I didn't see you accept the meeting invite, and I know there was some problems with the emails."

"Hi Clarie, I'm not really sure what this is about. What is this candidate for? I didn't have any idea that I had to do this interview today. Is it possible to get someone else?"

"We have a few people lined up for today. If I could rearrange it for 2:30 or for 3:00 today, would that be better?"

I see that I can't get out of this. "OK, I can do it now, I suppose. I'm going to be much busier later."

"OK great. The resume and feedback form is attached to the meeting invite. Thanks!"

So now I have to go interview someone. Looks like I'll never get those permissions tests done.

2:15

All right, so I met with the guy. He seems bright and eager. I quickly fill out the feedback form and mail it back to HR. And I get back to work. Setting up and running those permissions tests.

2:40

STROBE LIGHTS. SIREN.

"THIS IS YOUR FIRE SAFETY DIRECTOR. AT THIS TIME WE ARE CONDUCTING A MANDATORY FIRE DRILL, I REPEAT THIS IS A DRILL. PLEASE PROCEED TO THE DESIGNATED EXIT STAIRWELL AND STAND BY FOR FURTHER INSTRUCTIONS."

STROBE LIGHTS. SIREN. STROBE LIGHTS. SIREN. **STROBE LIGHTS. SIREN.**

"Oh, you've got to be kidding me! Today of all days! No way, I'm not going. I have too much to do. And I wasted enough time already."

Kathy, our building manager, doubles as the floor's fire warden. "Come on guys, you have to go, it's mandatory. No exceptions."

Sigh. "OK, I'm going, I'm going. At least it's not a full evacuation."

3:00

Back at my desk. Again. It's hard to make progress with so many distractions. But I'm making progress, even if it's choppy. There's just one more thing I want to experiment with here, and then I should go back to the lab and see how Raj is doing with those load tests.

Eric walks by. I'm afraid that he's about to embark upon one of his famous one-sided conversations. Eric is a good guy, and all, but sometimes he just doesn't know when to shut up.

"Hey, that was a great lunch with Harry, wasn't it? You know, he's such a great guy. Doesn't he look exactly like...oh, what's that actor's name? You know...from that movie..."

I better jump in and head him off at the pass. Otherwise I could get trapped here for half an hour.

"Hey Eric, no offense, man, but I'm really under the gun to get this testing done. Would it be OK if we chatted a bit later?

"Oh, hey, sure, no problem. Catch ya later."

3:50

I head over to the lab. Raj is there, he looks insane.

"Hey Raj, what's up?"

"I modified the script, and it's now pumping in five messages a second, instead of one message every five seconds. And I have it running on ten different machines. It's been running for a few hours now. But every once in a while, one of the workstations blows up."

"Blows up? What do you mean?"

"There! Did you see that?"

We see an error message pop up on one of the monitors over and over again, covering the entire screen with cascading pop up boxes.

"That can't be good."

Alright, we have to head over to the checkpoint meeting. Let's go, we'll just explain what we've done and where we are.

4:00

We all come in to the project manager's office. Me, Raj, Vlad, Eric, Kathy, and even Bob.

"So, where are we?" The project manager asks.

I review the test results that we have with them.

"After a rocky start because of environment issues, we've tested the basic features of the MMY feature, and we see that it works as we would expect it to. We also did some brainstorming with development to come up with additional relevant tests, which seem to work as we expected, too. We did discover that we needed permissionsing for MMY. Basic tests on that seem to work as well and I can show you exactly what we tested. We did find a minor bug, but I talked it over with Vlad and Eric, and we don't think it's a showstopper.

As far as capacity testing goes, we were able to generate load on the system, but Raj noticed that occasionally the workstations crash out of the blue. We're not sure why that is happening. We could use more time to do more tests."

"So, is the release blessed by QA, then?" The project manager asks.

Argh. How to answer this? Raj gives me a look. He knows what I'm about to say. Frankly, the project manager should know, too. We've talked about this before. But I will give him the benefit of the doubt.

I sigh, audibly.

"I don't mean to be facetious, but I don't know what that means. Surely you don't mean 'bless' literally. If you're asking me if I'm giving an official stamp of approval on this release, then I would tell you that I don't think I have the authority to do that on my own. I just gave you a summary of all the testing we've done, and our findings. But it is really not my decision alone. We should all discuss it, here, now. Is everyone in this room comfortable with the risks of releasing tonight?"

"So what do you think? Eric? Vlad?" asks the project manager.

"I think it's fine," says Eric. "Vlad?"

"Me too. You shouldn't worry about the crashes. You're testing on QA hardware. Production hardware is much better, they won't have those system issues."

"But we're still running the load tests" I say. "And we don't know for sure why it's happening."

"Production support needs lead time to set up the release. Let's put it in motion and tell them it's a Go. Meanwhile you continue testing in QA. And if we have to roll back, we'll roll back."

7:00

Raj and I hung around till it was released.

After the meeting, we went back to the lab to figure out what was going on. Turns out, there was a configuration setting that needed to be tweaked to handle the peak load. Once we figured that out, we told production support, who made the same change in prod during the rollout. I'm sure glad we did, otherwise we could have had a disaster tomorrow morning.

On the train ride home, I think back about the day. And what a day it was. It started with an interruption. How many interruptions did I have today? And which were productive ones, and which were unproductive? Which ones were I able to control, and which were out of my control?

At least it stopped raining.

GUERRILLA BUG REPORTING

by Brett LEONARD

Have you ever had an uncomfortable sinking feeling in your stomach after testing an application for just a few minutes? Perhaps you have more bugs than you could possibly report and retest in a reasonable time. Based on experience, you know some percentage of bugs will ping pong between you and the developers at least once before they are resolved, further extending your test cycle. Your release date is in serious jeopardy and you need a quick, effective, and creative solution.

Recently, I found myself in that exact situation. Our user interface development team did a complete rewrite of our user interface requiring my team to test every page and web element on our site. After just twenty minutes of brief testing, I found at least 15 bugs and I had just scratched the surface. I was looking for a quick to implement and creative solution. This article describes how we used Google Docs to report bugs, communicate with developers, and quickly stabilize our application.

Our development approach at Construction Software Technologies (CST) is Agile. At the time, we were using Visual Studio Team Foundation (TFS) to log our bugs. Logging a bug in TFS was a time consuming process. Occasionally, we adopted a more informal process to lower administrative overhead. We usually do this at the beginning of a test cycle for a new feature. This approach is effective but it comes at a cost. Bugs are undocumented and there is no traceability through the code. If the bug re-appears it looks like a new issue. This can lead to a different developer fixing a problem already fixed by another developer resulting in more instability. We try to focus on what makes sense in a given context rather than being slaves to a process. I did not know it but I was about to see an effective, low cost, organic process form right before my eyes.

My main goal was to stabilize the application as quickly as I could. I refused to be in a situation where I would test for an hour and then spend the rest of my day documenting bugs. That approach is not efficient or effective enough to achieve the goal. At first, I started to take notes in notepad, documenting bugs as I saw them. I planned to cover one area, email the developers and then move on to another area, gather more notes and repeat the process. As I tested, I became fearful that I was only covering a fraction of the application and it would take over a week of going back and forth with the developers before the application became stable. I realized that I needed my whole team actively testing the application with me. I needed a quick way of dividing the work that allowed for maximum coverage and minimal redundancy. That is when I turned to GoogleDocs.

Our group uses Google Docs for many of our test activities, so I am a big advocate of the product. I started with a minimalist approach; my initial spreadsheet had the following columns - page, responsible, and notes. I quickly took my notes and added them to the spreadsheet. Then I sent a link to the document to my team and instructed them to pick an area, note it in the spreadsheet, and start testing. At this point, I wanted my whole team testing the application identifying major issues. The strategy was to gather as much information as we could as fast as possible and then work out a way to communicate the issues to the developers.

During that day, one of the developers came by my desk and asked how things were going. I explained our strategy and the challenge of communicating test results. He went away and we continued testing. About a half hour later I got an instant message from him saying, "I heard you guys are using a GoogleDoc to track your testing, can you give me access to it so that I can start fixing the bugs you are finding?" Wow, I thought, what a novel idea! Give the developers access to the sheet so that they can fix bugs as we test; this could work. I immediately gave all the developers access to our spreadsheet and they began adding columns to suit their needs. Over the next few hours, the simple spreadsheet that I started morphed into a full-fledged real-time bug-tracking tool that had the exact information we needed to complete our goal.

During this time, our development manager expressed his concerns about this new approach. He pointed out that our bug count metrics would not be accurate. I convinced him that the most important goal was to stabilize the application. The best way to do that was to minimize administrative overhead and increase his team's visibility into our test results. He was still a little nervous but encouraged us to continue with our process. I promised him that once we were done, we would "collapse the sheet" and go back to entering bugs into TFS again.

To confirm fixes, we tentatively planned to have about three new builds each day. As the developers coded, they would mark the bugs they fixed on the spreadsheet. On the next deploy, the testers would immediately retest the fixed bugs and either close them or give feedback to the developers in the GoogleDoc. This allowed us to kick back bugs with practically no additional overhead.

After the first day, we found over 100 bugs. The next day we found another 50. The third and final day we found about 25. In three days we identified and fixed many high impact, high visibility bugs and brought stability back to the application. What started as a sinking feeling in the tummy ended with a huge success and new approach to turn around an application with swarming bugs.

If you choose to adopt this approach, here are some suggestions based on our experience:

- Start with a simple spreadsheet
- Encourage testers and developers to add columns as they deem necessary
- Use color coding to indicate whether issues have been fixed or still needs work: Green = Fixed, Red = Showstopper, Brown = Sent back to developer
- Go back to formal bug reporting when bug count becomes manageable again

About The Author

Brett Leonard *has worked in the software industry for more than 11 years as a tester, automation developer, project manager, and manager. He currently manages the software testing group at Construction Software Technologies, also known as iSqFt, where he supports new development and maintenance of a business-to-business networking platform for the construction industry. Brett can be reached via email at brettlleonard@gmail.com.*

COMING TO TERMS WITH TEST AUTOMATION:

A Dialogue Between Michael Larsen and Albert Gareev

It's becoming a more common occurrence with software testers. Automation isn't just a nice thing to dabble in; it's actively becoming a differentiator for companies. Look at the job postings – a large percentage of them are focused on testers who code, developers who test, or variations on those themes in between. So testers develop coding chops, they practice development at home or with other groups (meet-ups, coding dojos, hack nights, etc.). So testers are preparing to dive head first into this wild and wooly world of automated software testing.

STOP! Rewind! Take a deep breath.

Who's making these decisions? What are these projects going to address? Do you know? Does anybody know? I felt a fair amount of frustration at some of the projects I'd participated in over the years, and often felt that they had no real direction, no clear goal or focus, and that it was just automation for the sake of automation. With this in mind, I thought it would be worthwhile to have a chat with someone who knows quite a bit about automation and writing automation frameworks, but also has a fairly solid hand in the real world of active software testing. To this end, Albert Gareev agreed to talk with me about this challenge and how we can effectively address it.

MICHAEL: "So Albert, here's the thing. My company has asked me to help develop a way to automate all of our functional tests. I've made a full tour of the features that we will need to cover and I'm excited to get started in this automation adventure."

ALBERT: "Well that's good, Michael, but before you dive in, there are some things you might want to consider first. Test automation is a controversial topic. To begin with, a *test* is never "automated", so we're starting from a bad premise right there. What really happens is that a pre-scripted sequence of instructions is run. This more or less can imitate manual operations performed by a human. Any active thinking and learning is completely missing, as well as many other things that skilled testers do subconsciously."

MICHAEL: "OK, Albert, I see the distinction, but if that's the case, does it really make sense to automate… sorry, pre-script sequences of instructions? If we are not going to be actively thinking and learning, what's the benefit of this?"

ALBERT: "Michael, the benefit is that you really want to be continuously engaged in an active search for new problems and new risks. Automation, in turn, can be of service to you in the process of verification and confirmation that previously implemented requirements are still met, and known risks do not reappear. Furthermore, there are critical product risks that can only be discovered through high-volume, prolonged, stochastic, or beneath-the-GUI interactions, or even through a mixture of mentioned types.

MICHAEL: "So there's no real way to get around it, is there? It's important, but it seems that people are at cross-purposes when they try to automate feature testing. It seems to me that there are a lot of factors at play here. The importance of each factor might vary, depending on the context of a project."

ALBERT: "That's true, but there is a way that we can evaluate many of these factors and put them together to help guide us as we make this evaluation. There are a few heuristics that we can use to help us in this process, and a simple mnemonic to help us remember them. To evaluate a potential automation project, it helps to think of the TERMS of the project."

MICHAEL: "TERMS?! All right, Albert, you've piqued my interest. What is TERMS, and how can we use it?"

T Tools and Technology

ALBERT: "The first heuristic is tools and technology. We have to think first, how are we going to make this project work? The platform will either offer options or constraints as to the testing that can be performed. We will have different requirements for a mobile app than we will for a desktop application, or for a web application."

MICHAEL: "So Albert, let's see if I can set this up. Suppose I am testing a web application that is accessible by both traditional web users and mobile users. My first question would be if we can support and work with the technologies we want to test with the tools we have?"

ALBERT: "Correct, we would need to know – are the application's technologies well supported by the tool? Actually, this is part of a bigger question. We need to evaluate recognition, connection, interaction, and reliability of those operations. From there, we would create a working code for each operation. Then we would test it on a few samples from the application. For example, see if the same instructions you used to interact with one GUI table can be reused with another kind. You would also want to test behavior of your automation code with slower and faster performance of the application."

MICHAEL: "OK, I get it. This way we'd want to make sure that we were able to address all of the interaction points. Is this enough for us to say we've met the requirements for this heuristic?"

ALBERT: "Not quite. We'd also want to make sure that we were able to handle a number of unusual actions. Consider loading unusual objects – custom objects, non-existing objects, hidden and/or disabled objects, and objects that can change their properties at run-time. It often happens that certain problems in GUI recognition and interaction can be resolved from the application's side. If that is the case, we'll need to assess how significant those changes will be."

MICHAEL: "So you're saying that these changes might require a one time modification, or they might require continuous maintenance from the development team, depending on the area under test?"

ALBERT: "Exactly. Furthermore, we also need to take into account any extra testing effort that would be required to investigate whether the changes impact the applications functionality or compromise security. Last but not least is to look forward; if (or when) the application's technologies evolve further how will the answers to all of these questions change?"

E Execution

MICHAEL: "So Albert, you are familiar with the old statement any repetitive task should be automated. What do you think, is that a good place for us to start, just gather together all of the most repetitive areas and automate them? Seems like it would be a solid win to me."

ALBERT: "Michael, you are right, that is a popular statement, and it's just as often wrong as it is right. Let's step back and consider some of these steps:

- Are we looking at a manual, repetitive process, or are we looking at an area that is really a sapient[1] process?
- Does it look repetitive only on a high level, while actually it is a highly cognitive work, which can't be done without human judgment?

Even if the task does not involve focused thinking, it may only appear repetitive. It's possible that logic and data may be mutating with every transaction. Automation under these circumstances is a much more complex task".

MICHAEL: "I think I see your point here, Albert. We run tests to help reveal and gather information about the product. Automation, by its very nature, has extremely limited observation capacities. The question of a trade-off is critical; does automated script execution give some advantages over manual execution?"

ALBERT: "Yes, Michael, it does. For example, a common automation tactic is the creation of a single transaction as a "test case". This reduces the scope and quality of testing, in comparison to skilled human exploration. Running such an automation script hundreds of times, however, creates a new volume test, which may help revealing memory management issues or race conditions problems in the application. Executing the script concurrently on multiple machines creates a load or stress test. Execution of scripts creates an opportunity for testers to observe and identify problems. But execution by itself does not find any problems."

AUTOMATION ISN'T JUST A NICE THING TO DABBLE IN; IT'S ACTIVELY BECOMING A DIFFERENTIATOR FOR COMPANIES. — MichaelLARSEN

MICHAEL: "So if I'm following what you are saying, by continually evaluating frequent execution of complex scripts, we are required to put an emphasis on robustness and reliability, right? The point is, we have to spend considerable effort creating and maintaining these automated test scripts. Information that these automation scripts are meant to gather have to be properly collected, stored, and reported. And the reports must be user-friendly, and at the same time detailed enough for a comprehensive analysis."

ALBERT: "That's right. If implemented, these automation assisted testing activities then become an integral part of the project. They also introduce new risks. How much can you rely on a prompt execution when requested? What is your back-up plan if you can't use them as expected?"

R Requirements & Risks

MICHAEL: "You are right, we haven't even addressed the risk components associated with adding in an automation project. Too many requirements, not detailed enough requirements, frequently changing requirements – all of these are never ending headaches for testers."

ALBERT: "Well, that might be a bit dramatic, but yes, there is a lot that can change and that we have to be aware of. But people can learn, analyze, and adapt. Also, there is an arguable chance of creating useful automation that verifies expected results. Actually creating a script that can explore, investigate, and report about possible problems is beyond imagination."

MICHAEL: "I agree, it would be a bit of a stretch to think that scripts could reliably do all that. It seems to me that, if we have stable requirements, most of the potential problems will be found *during the creation* of automation scripts. Of course the same could be said that those same problems could have been found during a regular testing cycle. With unstable requirements, any verification-oriented automation will demand an endless maintenance. The possible problems from this would require a discussion all by itself. So we should be asking what product risks automated scripts will be able to address. We should also consider those areas that automation won't be able to address. If some risks are not addressed with automation, will human-engaged testing be required? What if all risks can be covered with nearly the same effort?"

M Maintenance

ALBERT: "Michael, these are all important questions, but there's an area that doesn't get the level of consideration that it deserves, and that is maintenance. After an initial and expensive development effort creating automation scripts, it would be desirable to use and maintain those scripts at minimal cost. In general, the maintenance effort is distributed through the following components of the automation suite:

- Source code (most expensive)
- Test logic (moderately expensive)
- Test data (least expensive)

It's important to note that, what we might be considering as "cheap maintenance" could become very expensive if you have a large volume of data or your data sets change very frequently."

MICHAEL: "Sure, that makes sense, but what if we are looking at changing up the application's front eng (i.e. the GUI)? It's been my experience that many tests that rely on the GUI to be configured a certain way can have major problems when configuration changes. So it will be necessary to explore the front end when these changes are made, and then re-write many of the tests? And since I've already explored those areas of the application (that is – already tested) what's the point of spending time fixing and running scripts?"

ALBERT: "Yes, that's a big question to consider, but there are others. Many tests can be performed by bypassing the GUI altogether. That opens up other issues. Those "hooked" tests may not give the answers that the developers are after (testing under the GUI will test functionality, but it will not give any indication of the actual flow the user will see). I think it's important to consider that, to do debugging and updating existing code, we'll need to have people with programming skills and experience. Look at it this way, if the testing logic is baked into the source code, the task of maintaining test logic becomes as expensive as it is to perform source code maintenance. What if test data were hard-coded? This also brings us to an all to common occurrence; what if updating / fixing automation scripts take more time than performing manual, engaged sapient testing of the same areas of the application?"

S Security

MICHAEL: "So Albert, we have covered all of the main areas, I think. What does the 'S' stand for?"

ALBERT: "It stands for Security. It's listed last because it may or may not be applicable in your evaluation. Still, even if the application we are testing has no security policies, it will be important to determine that the automation tool itself does not impact or compromise corporate security settings and policies."

MICHAEL: "That's a good point Albert. What if our testing tools require disabling of the firewall, or granting administrator privileges? Or maybe even more pressing, what if the tool reads username/password data, allowing access to sensitive information, and stores this information on a publicly available network folder?"

ALBERT: "Those are important considerations. Here are some additional ones. If the application has security features itself, it introduces extra challenges in automation – and more questions to consider in your evaluation. For example, connecting over secured Internet connection protocol (HTTPS) might not be supported by the tool. Is turning it off such a good workaround? How much could it obfuscate the performance test results? If your automation relies on API calls, do you plan to test afterwards that the final build does not allow using any of these backdoors?"

Where To Begin?

This is a lot to think about when we start down the road to consider if automation is right for a particular project. Automation is a service to testing. I usually start from the assessment of testing needs (execution and requirements) and risks. Evaluation from those angles can tell us if automation potentially makes sense. An application's technologies will narrow down a list of tools you can possibly use, and expenses associated with a tool may affect the decision whether it's feasible to do.

Once a tool has been selected, it's time for hands-on practice and exercises. These may often reveal technical challenges and limitations. Later the code from the exercises might be used for the building of "proof of concept" scripts. Of course, if any changes in the application were made in order to build a "proof of concept", analysis of those changes and testing to explore possible impact on security or application's functionalities must also be performed.

Testing needs and the available timeframe will have a direct impact on the automation approach. However, maintenance requirements must also be considered before you get started.

Defining our automation TERMS will help us to better assess capabilities and also provide a more solid grounding of our expectations. It's entirely possible that, after considering these steps, you might decide not to invest into a particular kind of automation or in automation at all. If, however, you do decide to proceed, you will be much less likely to feel disappointed in the end.

1. James Bach, "Sapient Processes" http://www.satisfice.com/blog/archives/99

About The Author

Michael Larsen *is a lone tester with SideReel.com in San Francisco, CA. He is a brown belt in the Miagi-do School of Software, an instructor with the Association for Software Testing, facilitator for Weekend Testing Americas and the producer ofSoftware Test Professionals "This Week in Software Testing" podcast. He can be found on Twitter at @mkltesthead and blogs at http://mkl-testhead.blogspot.com.*

Albert Gareev *is a dedicated testing and test automation professional, technical lead and a hands-on consultant. He combines passion for testing as investigation and learning with an automation approach helping to reach observation levels beyond "naked eye and hand" in complex, large-scale systems. Albert frequently writes in his blog (http://automation-beyond.com/) and occasionally tweets as @AGareev.*

TROUBLE TICKETS ARE YOUR BUSINESS

The **Insular World** of Software Development

When your team sits down at a project kickoff, look around the table and see who determines your software's purposes and workflows. You will probably see a project manager, some developers, some quality assurance people, probably a sales-oriented client-facing person with some faddish title, and perhaps a business analyst if your organization is big enough. **Who is missing from this equation? *The end user.***

by Brian J. **NOGGLE**

About The Author

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.*

No doubt, your organization will do some requirements gathering that will include talking to a user, or at least talking to a manager at the user's employer or benefactor in some software development fashion. For example, some companies build internet tools or develop web sites based on their ideas for what their users might need. However, unless your organization sports a user interface testing lab with cameras tracking user actions and analysts trained in auguring UI design, your understanding of what the user will do and how he will do it remains incomplete and sometimes completely fictional.

One group in your organization (or in your customer's organization) can yield great insight into the field conditions of your software and the sometimes crazy things your users do with your software. The technical or customer support staff at your organization talks to your users all the time. If you're not mining this rich vein of data as you develop, or at the very least, test the software, you risk missing information that would make your tests, and your software, better.

Who Are Customer/Technical Support?

Different organizations handle customer or technical support in a myriad of ways, but most have someone to answer the telephone or to field emails from users who have questions or who encounter some problem with your software or their understanding of it. Hopefully, your organization has a trouble-ticket system where the support people enter information about the issues, including details about where in the software the problem happened; what the user was doing; and where the user was running the software. Typically, support then kicks these tickets to field staff for resolution; to developers for bug fixes; or to support staff to help the customer immediately. The appropriate parties take action and then close the ticket.

Different organizations have varied processes. Back-end project-based (as opposed to product-based) development shops might not have support at all if they merely write code for a paycheck and turn it over to a client who skins or badges the software and sells/rents it/gives it to its users. In the latter case, someone at your client handles support duties, and you should contact that person to mine the information in those trouble tickets.

What Support Can Teach You About Users

Trouble tickets can teach you a lot about your users. When you and your organization are steeped in the software, enmeshed in your understandings of what it should do and how it should do it, you spend ten hours a day looking at its interfaces or code on your computer. That's your job.

Users, on the other hand, have another job to do, be it waiting on customers in a retail store, logging information in a research lab, checking out books at the library, or a host of other jobs. That is their business. Their industries have not only languages of their own, but also mindsets of their own which come with certain expectations of software behavior. Your user scenarios, requirements, and other project documentations only capture glimpses of their expectations.

At points where your software does not match those idioms and mindsets, users will have questions or misunderstandings. If your software does not use the right terms, the users won't understand. For you in the development world, clients and libraries have different meanings than they do to an actual librarian, who refers to library users as "clients." If your screens do not match the written forms your software automates or your workflow does not match the process in place at your client site, your users won't recognize the happy path workflow that your organization has tested out ad infinitum through the automated tests, the smoke tests, and demos. If your software does not perform adequately, interrupting the user's regular duties, the user might try to find alternative workflows through your software or shortcuts that your software was not intended to support.

Reviewing open and closed trouble tickets should highlight any nomenclature disparities. Additionally, trying some of the shortcuts that your users discover, whether those shortcuts succeed or uncover a defect, should give you new directions for your tests. At the very least, the tickets of actual user shenanigans provide anecdotes to riposte when the developer claims, "No user would do that!"

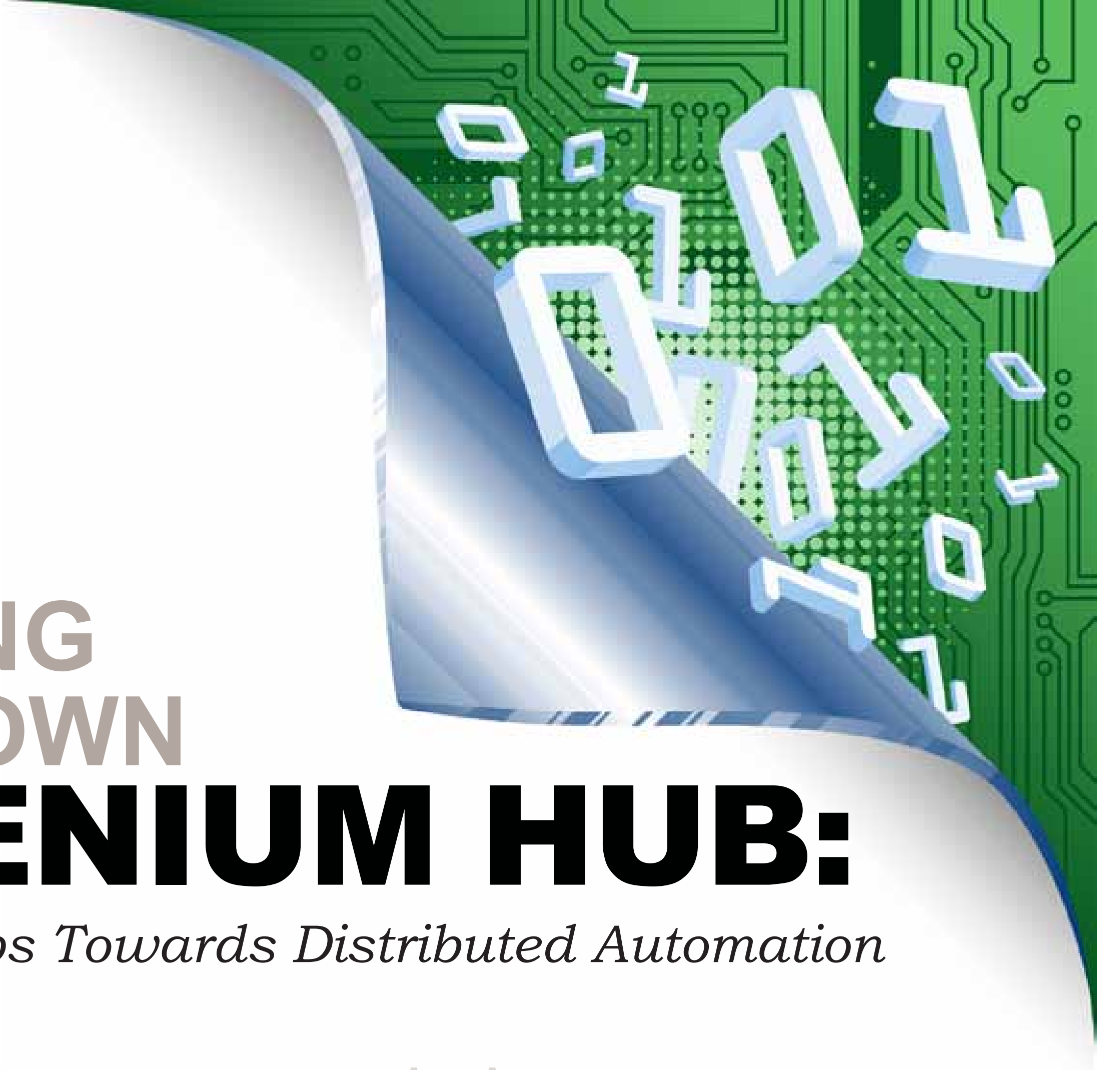
What Tickets Can Teach You About User Environments

Unless you're like Apple and lock down hardware and software environments for your product tightly, your users are going to install your software in a wide variety of environments or use a myriad of browsers with different plugins and extensions installed. When your organization began this project, you guessed at a set of environments to test and to support. Trouble tickets can identify combinations that cause trouble for actual users so you can include them in your testing or to explicitly deny support.

For example, when I was working for an interactive agency in 2005, we built consumer-facing websites with data collection forms and contests. A large number of client support issues arose when webTV users tried to enter the contests. They filled in their information, but when the users clicked "Enter now!", submission failed due to the lack of JavaScript support in that particular browser. Our organization didn't decide to support webTV. Instead, when the browser string indicated that browser, the website displayed an alternate message advising the user to try a different browser. The number of user complaints, at least those expressed to our client's customer support team, declined, and the webTV user's experience with those web sites improved slightly.

Conclusion

This article can only give a couple of examples of the sort of intelligence you can ferret out of your trouble tickets to improve your testing practices and, ultimately, your software product. It can only encourage you to review the customer contact documentation your organization captures. Trouble tickets might inspire you to find new ways of thinking more like end users who must deal with your software day in, day out.



RUNNING YOUR OWN SELENIUM HUB:

The First Steps Towards Distributed Automation

Selenium WebDriver is a great open source tool that is being used by thousands of companies worldwide. While several people are satisfied by the features that the Selenium IDE offers, others want more flexibility by writing their own scripts using programming languages such as Java, Python and Ruby.

by Felipe **KNORR KUHN**

That's because, when you take control of the Selenium API and combine it with a powerful testing framework, you can get, as a result, a robust data driven, distributed, multi-platform regression suite.

Before the release of Selenium 2.0, configuring a grid wasn't a very straight forward task. Now, all the needed parts are packed into a single file and it no longer depends on ANT to launch the server and the nodes, which makes it easier to get your environment up and running.

Some companies (including Sauce Labs, the company founded by Jason Huggins, the original Selenium author) are even offering services that provide ready to use, elastic grids, so you don't have to go through the hassle of configuring everything on your own. However, before you jump into the cloud testing bandwagon, you might consider reading this article so you can learn the basics and understand what goes on under the hood.

In this article, we are going to setup a very basic distributed environment, composed of the following elements:

- An Ubuntu 11.10 server to run the hub, which will listen for incoming connections from the nodes and the test invokers
- A Windows XP machine that will serve instances of Firefox 8.0.1 and Internet Explorer 8
- A Windows 7 machine that will serve instances of Firefox 3.6 and Internet Explorer 9

Before we start configuring the hub and the nodes, download the latest version of the Selenium Server at the official Selenium website[1] on all machines. As of the writing of this article, the latest version was 2.13.0.

Now, we are going to launch the hub so it will listen for incoming connections from the node machines and the client machines that will request the browsers to be run against the script. Open a terminal window and run the following command:

- java -jar selenium-server-standalone-2.13.0.jar -port 4444 -role hub -nodeTimeout 600

If everything goes as expected, you will see a message saying that the grid server is up.

Running only the hub is not very useful, as we will not be able to launch any scripts using the hub alone. We need to register at least one node to the server. Let's start with the Windows XP machine, which will provide an Internet Explorer 8 instance and up to two Firefox 8.0.1 instances.

One of the lesser known features of the Selenium WebDriver grid configuration (which I am glad for my friend and coworker Iraê Carvalho[2], who has pointed this feature out[3]), is that you can use a JSON file (named "selenium-node-ie8-ff801.cfg.json" in our case) to configure the node, which makes it very easy to setup and maintain, as you can see on Figure 1.

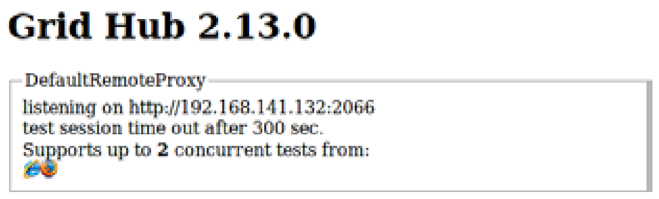
```
{
    "class": "org.openqa.grid.common.RegistrationRequest",
    "capabilities": [
        {
            "seleniumProtocol": "WebDriver",
            "browserName": "internet explorer",
            "version": "8",
            "maxInstances": 1,
            "platform": "WINDOWS"
        },
        {
            "seleniumProtocol": "WebDriver",
            "browserName": "firefox",
            "version": "8.0.1",
            "maxInstances": 1,
            "platform": "WINDOWS"
        }
    ],
    "configuration": {
        "port": 2066,
        "register": true,
        "host": "192.168.141.132",
        "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
        "maxSession": 2,
        "hubHost": "192.168.141.135",
        "role": "webdriver",
        "registerCycle": 5000,
        "hub": "http://192.168.141.135:4444/grid/register",
        "hubPort": 3333,
        "remoteHost": "http://192.168.141.132:2066"
    }
}
```

Figure 1 – Node Config

After we are done with configuring the node (please note the IP addresses for the hub and the node), it's time to launch the node and register it with the hub with the following command:

- java -jar selenium-server-standalone-2.13.0.jar -role webdriver -nodeConfig selenium-node-ie8-ff801.cfg.json

A message saying that the node is being registered to the hub will appear. To make sure that the hub has registered the new node, load the grid console at http://<hub-ip>:<hub-port>/grid/console and notice that it is listed there, as seen on Figure 2.



Grid Hub 2.13.0

DefaultRemoteProxy
listening on http://192.168.141.132:2066
test session time out after 300 sec.
Supports up to 2 concurrent tests from:

Figure 2 – The Selenium Hub Console

Configure as many nodes as you want, preferably with different browsers and platforms, such as Windows 7 with Firefox 3.6 and Internet Explorer 9 (as suggested in the beginning of the article), Linux or Mac OS X with Firefox 3.6, and so on.

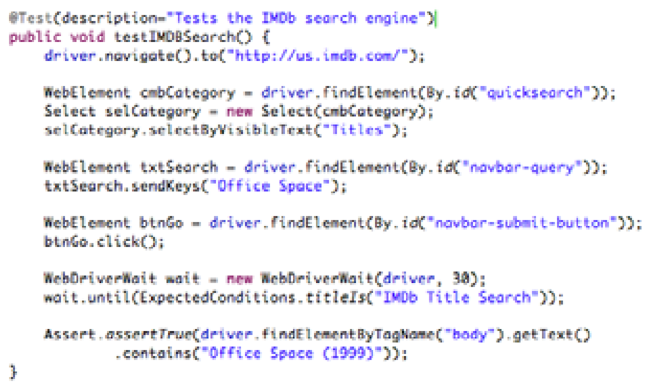
Now we will need an automation script and a way to select in which browser/platform combo we would like to run it. That's where the flexible testing framework kicks in. I'm happy with TestNG in the Java world, so I am going to use it to illustrate a simple IMDB (internet movie database) example here.

First, we create a new test class with a @BeforeTest method that will read the TestNG configuration file and instantiate the RemoteWebDriver object considering the platform, browser and version we want to run as seen on Figure 3. The server URL field can also be loaded from the configuration file. How about taking that as an exercise?

```
@Parameters({ "platform", "browser", "version" })
@BeforeTest(alwaysRun = true)
private void setup(String platform, String browser, String version)
        throws MalformedURLException {
    DesiredCapabilities caps = null;
    if (browser.equalsIgnoreCase("Internet Explorer")) {
        caps = DesiredCapabilities.internetExplorer();
    }

    if (browser.equalsIgnoreCase("FireFox")) {
        caps = DesiredCapabilities.firefox();
    }

    if (platform.equalsIgnoreCase("Windows")) {
        caps.setPlatform(org.openqa.selenium.Platform.WINDOWS);
    }

    if (platform.equalsIgnoreCase("Linux")) {
        caps.setPlatform(org.openqa.selenium.Platform.LINUX);
    }

    caps.setVersion(version);

    String remoteServer = "http://192.168.141.135:4444/wd/hub";

    driver = new RemoteWebDriver(new URL(remoteServer), caps);
}
```

Figure 3 – The Test Setup Method

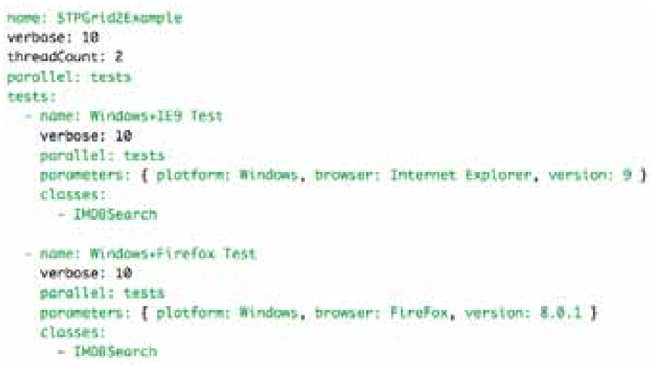
Next, we write the script that will go to the IMDB website, select "Titles" from the search criteria dropdown, fill in the search box with the movie "Office Space" and then search for the movie, checking whether or not it appears in the search results (Figure 4):



```
@Test(description="Tests the IMDb search engine")
public void testIMDBSearch() {
    driver.navigate().to("http://us.imdb.com/");

    WebElement cmbCategory = driver.findElement(By.id("quicksearch"));
    Select selCategory = new Select(cmbCategory);
    selCategory.selectByVisibleText("Titles");

    WebElement txtSearch = driver.findElement(By.id("navbar-query"));
    txtSearch.sendKeys("Office Space");

    WebElement btnGo = driver.findElement(By.id("navbar-submit-button"));
    btnGo.click();

    WebDriverWait wait = new WebDriverWait(driver, 30);
    wait.until(ExpectedConditions.titleIs("IMDb Title Search"));

    Assert.assertTrue(driver.findElementByTagName("body").getText()
            .contains("Office Space (1999)"));
}
```

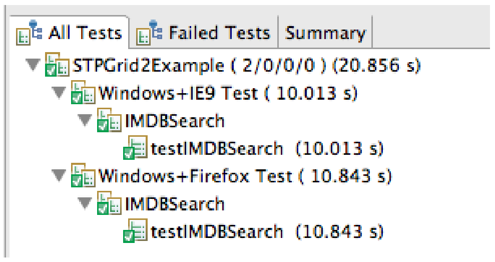
Figure 4 – The Automation Script

The TestNG configuration file (in YAML format) used in the example is shown on Figure 5, and the results of the execution (launched from the Eclipse IDE) can be seen on Figure 6.



```
name: STPGrid2Example
verbose: 10
threadCount: 2
parallel: tests
tests:
  - name: Windows+IE9 Test
    verbose: 10
    parallel: tests
    parameters: { platform: Windows, browser: Internet Explorer, version: 9 }
    classes:
      - IMDBSearch

  - name: Windows+Firefox Test
    verbose: 10
    parallel: tests
    parameters: { platform: Windows, browser: FireFox, version: 8.0.1 }
    classes:
      - IMDBSearch
```

Figure 5 – The TestNG Config File



```
▼ STPGrid2Example ( 2/0/0/0 ) (20.856 s)
  ▼ Windows+IE9 Test ( 10.013 s)
    ▼ IMDBSearch
        testIMDBSearch  (10.013 s)
  ▼ Windows+Firefox Test ( 10.843 s)
    ▼ IMDBSearch
        testIMDBSearch  (10.843 s)
```

Figure 6 – The TestNG Execution Results

This article was intended for educational purposes only and should not be taken as the definitive implementation. Feel free to refactor the code or adapt the ideas including the programming language and frameworks used to suit your needs. As an extra exercise, you can add more machines to the hub and play with the parameters to run the script on different configurations.

I hope it is enough to get you started!

1 http://seleniumhq.org/download/

2 http://tech.irae.pro/

3 http://opensourcetester.co.uk/2011/07/06/selenium-grid-2/

About The Author

__Felipe Knorr Kuhn__ found his passion for software testing when he was a child and used to look for bugs in videogames. He currently works as a software engineer at Yahoo! Brazil. Felipe is a strong believer of the benefits of collaboration, and can be reached in many Brazilian and international software testing and open source communities.

www.zelger.org
"OUR ASTEROID TRACKER ONLY MENTIONED A NEAR MISS UNTIL YESTERDAY, WHEN I INSTALLED THE LATEST PATCH"
BOSS

Who is actually testing near earth object tracking software?