

The **INTERNATIONAL** Monthly On Software Testing

Tea-time with Testers

Jerry Weinberg

The Golden Key (Part 2)

Elisabeth Hendrickson

Beware The Hero !

Gojko Adzic & David Evans

Visualising Quality

Karen Johnson

"What if" in Mobile Testing

Anurag Khode

Getting Started with Mobile Apps Testing

Anne-Marie Charret

Career Management for Software Testers

Michael Larsen

Pairing with an Expert

Joel Montvelisky

Choosing Correctly What NOT to Measure

Anuj Magazine

Software Testing and Art of Staying in Present

T Ashok

Musings & Inspirations :
Learning from Non-S/W Testing Disciplines



EACH ISSUE IS
A **GREEN** ISSUE

JULY 2011 | Year 1 Issue VI | www.teatimewithtesters.com

© Copyright 2011. Tea-time with Testers. All Rights Reserved.



Dear All,

"Despite of all hurdles, you must keep going" is the mantra of Mumbai and following the same, hereby I offer you the July issue of Tea-time with Testers.

In order to provide you yet another bigger platform, am happy to announce our collaboration with world class Testing Communities like Quality Testing and Mobile QA Zone.

Regarding our Teach-Testing Campaign; YES! We are going good and am pleased to get increasing support for it. You will get to read some of our supporters' responses who belong to different corners of the World.

Am also happy for having received so many letters from our readers; appreciating our work, passion for Software Testing and what has delighted me most, is their willingness to join/represent Tea-time with Testers, without being asked. I sincerely thank all of them and will make sure to provide them with opportunities to grow with us.

Once again, many thanks to our content authors for devoting their valuable time and making this issue of Tea-time with Testers worth enjoying.

Well, you all must be aware of the unmatched and exhilarating spirit of our Mumbai city which also happens to be the birthplace of "Tea-time with Testers"

The news of serial bomb blasts in Mumbai that happened lately was quite dolorous. Having our roots in this promising city, we strongly condemn this attack and also similar kind of mishaps happening across the globe. I hope that peace will be there established forever.

With lots of hopes for peaceful tomorrow, I dedicate this issue of Tea-time with Testers to the World-Peace.

Enjoy Reading! And have nice Tea-time!

Yours Sincerely,

 **Lalitkumar Bhamare**



Created and Published by:

Tea-time with Testers.

Hiranandani, Powai,
Mumbai -400076
Maharashtra, India.

Editorial and Advertising Enquiries:

Email: teatimewithtesters@gmail.com
Pratik: (+91) 9819013139
Lalit: (+91) 9960556841

© Copyright 2011. Tea-time with Testers.

This ezine is edited, designed and published by **Tea-time with Testers.**

No part of this magazine may be reproduced, transmitted, distributed or copied without prior written permission of original authors of respective articles.

Opinions expressed in this ezine do not necessarily reflect those of editors of **Tea-time with Testers** ezine.

QuickLook



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

Beware The Hero ! -13

Visualising Quality- 17

S/W Testing & The Art of Staying in Present-21

In the School of Testing

Finding Your Niche in Software Testing-27

"What If?" in Mobile Testing - 31

Pairing with an Expert - 35

Getting Started with Mobile Apps Testing: - 37

Testing Intelligence: Choosing Correctly What NOT to Measure - 42

T' Talks

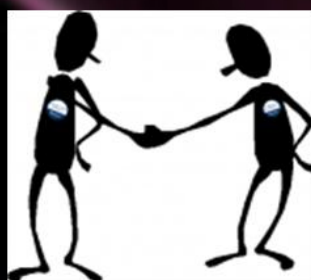
Musings & Inspirations: Learning from Non S/W Disciplines- 47

Tool Watch

Testing Puzzle – S.T.O.M. Contest

Our Testimonials

Family de Tea-time with Testers



Finding A Niche

Testing Puzzles

by Sebi

Crossword
by



Musings

What's making News?

- find out the latest happenings in the technology world

image: www.blofoto.com

Tea-time with Testers collaborates with **Quality Testing** and **Mobile QA Zone**



PRLog (Press Release) – Jul 17, 2011:

After successfully creating a remarkable impact in Global Software Testing community, **Tea-time with Testers** magazine has now decided to tie its bonds with world class Testing Communities like **Quality Testing** and one of its own kind community for Mobile Application Testing i.e. **Mobile QA Zone**.

Tea-time with Testers is known for the platform it provides to testers across the globe, quality of its content and unique offerings facilitating its readers to let them know about various dimensions of Software Testing. With launch of its June 2011 Issue; this magazine has reached up to 62 countries in the world.

By collaborating with Quality Testing and Mobile QA Zone, Tea-time with Testers is aiming at providing a much bigger platform to its readers and offering them with yet more quality stuff.

About Quality Testing:

Quality Testing is a leading social network and resource center for Software Testing Community in the world, since April 2008. QT provides a simple web platform which addresses all the necessities of today's Software Quality



beginners, professionals, experts and a diversified portal powered by Forums Blogs, Groups, Job Search, Videos, Events, News and Photos.

Quality Testing also provides daily Polls and sample tests for certification exams, to make tester to think, practice and get appropriate aid.

About Mobile QA Zone:



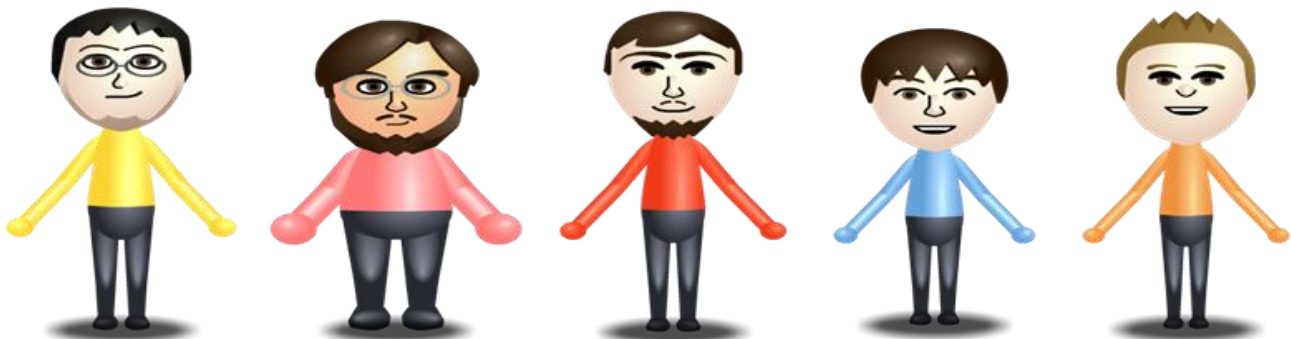
Mobile QA Zone is a first professional Network exclusively for Mobile and Tablets Apps Testing.

Looking at the scope and future of mobile apps, Mobiles, Smart phones and even Tablets, Mobile QA Zone has been emerging as a Next generation software testing community for all QA Professionals.

The community focuses on testing of mobile apps on Android, iphone, RIM (Blackberry), BREW, Symbian and other mobile platforms.

On Mobile QA Zone you can share your knowledge via blog posts, Forums, Groups, Videos, Notes and so on...

Click on the Logos to know more about



our allies

[Back To Index](#)



Look Who's Rising



Six Months

5000+ Readers

62 Countries

ONE Magazine

TEA-TIME WITH TESTERS

Subscribe here Right Away to get our all Issues for FREE

Tea & Testing



with

Jerry Weinberg

The Golden Key (Part 2)

Please check our June issue for Part 1

And then they would loop back to the beginning, raising the temperature a few degrees.

I thought that the problem-solving would go better if I could cool things down, but all I was hearing was "yes-you-did-no-we-didn't," back and forth. I decided to attempt to establish some facts that were not a matter of opinion, so I asked for the original requirements document. Both Penny and Jeff seemed a bit stunned by this reference to data, then Penny recovered and said, "Yes, that will prove my point."

"No, it will prove my point," Jeff countered. "Good idea, Jerry. Now we'll see whose fault this is."

I was a bit surprised at how readily they each found the document. (Lots of my clients seem to lose requirements documents once a project is under way.) Jeff got his open first, and placed his index finger on the following key line:



The Catalog Department should deliver component pricing data by 1 February to the IT Department.

I thought Penny would find some other statement to "prove" her point, but a few moments later, she had her copy open to the same page, upon which the same sentence was highlighted in DayGlo pink. "There."

She said, triumphantly. "There's my proof. We never promised to deliver that data that early."

"Yes you did. It's perfectly clear, right there. Should deliver by 1 February."

"Exactly," Penny countered. "It doesn't say we will, but only that we should. And we did try. But you computer people apparently don't appreciate the difficulty of getting every single one of those prices signed off by every person involved."

Well, I eventually got things cooled down, and we moved from blaming to problem-solving, but not before I extracted a promise from both parties to attend a little workshop I designed for them. I designed the workshop because I didn't want to have to come back the next summer when they ran into the same problem—a lack of understanding of the ambiguity of the English language. The following are some excerpts from that workshop:

Should

I started the workshop with focus of their original problem, the nasty little word, "should." Jeff read the original statement as The Catalog Department [must] deliver component pricing data by 1 February to the IT Department.

Penny, however, interpreted the "should" differently, as The Catalog Department [will make every effort to] deliver component pricing data by 1 February to the IT Department.

What I taught them was a safer meaning, of "should" would be "probably won't," so the sentence reads,

The Catalog Department [probably won't] deliver component pricing data by 1 February to the IT Department.

"Oh," said Jeff, "if I'd realized that, we could have designed the project differently. Could Catalog have delivered parts of the pricing data by February 1?"

"Sure," said Penny. "We actually had about 90% of it by then, but that last 10%— mostly new items—took all the work." "Ah. If only we'd known. We didn't need the entire table to proceed. Okay, next time we'll just let you know what we really need."

Just

Jeff had given me the perfect opening for the next lesson. "Sorry, Jeff," I said. That won't do."

"Why not?"

"Because you've managed to sneak in another one of those discounting words."

"What word?"

"Just." I went to the whiteboard and wrote what he said:

"Next time we'll just let you know what we really need."

"Now, what's the difference between that sentence and this one? I wrote:

"Next time we'll let you know what we really need."

"Well, it's the same thing, isn't it?"



Penny chimed in. "I get it. The 'just' makes it sound like there won't be any problems. It discounts the difficulty."

"Precisely. It's what I call a 'Lullaby Word.' Like 'should,' it lulls your mind into a false sense of security. A better translation of 'just' in Jeff's sentence would have been, 'have a lot of trouble to.'"

"I get it," Jeff said, coming to the board and snatching the marker from my hand. Then he wrote:

"Next time we'll [have a lot of trouble to] let you know what we really need."

"You know," he sighed, "I wish we'd had this little lesson last month. My second-best analyst up and quit on me, and I didn't see it coming. But a few weeks before, he told me, 'We haven't managed to hire another analyst yet, so I'm just working 80 hours a week until they do.' I should have heard him saying,

'We haven't managed to hire another analyst yet, so I'm [having a lot of trouble] working 80 hours a week until they do.'

He was trying to tell me that he was overloaded, but the 'just' lulled me into discounting his message. And, because I didn't hear him, he finally quit. Darn!"

Soon

Penny looked thoughtful. "I know another Lullaby Word that got us into trouble."

"What's that?" Jeff asked.

"You remember when we didn't have the prices ready on February first, and you asked me when we would have them?"

"Sure, but I can't remember what your answer was."

"That's because it was a Lullaby. I said, 'Soon.' And what that means is..." She looked at me, and I nodded.

"I think it meant, 'I don't know, but don't keep bothering me.'"

"That's usually a pretty good translation," I confirmed.

Very

"Actually," Jeff chimed in, "what you said was 'very soon.'"

"Oh, great!" Penny said. "And what did that mean?"

"Adding 'very' is like adding a sleeping pill to the lullaby. It makes it even more certain that it's going to be a long, long time. Maybe never."

We spent a bit more time on the subject of lullaby words, with examples such as Only: It's only a one line change. [That is, I didn't think much about what could go wrong.]

Anything: I didn't change anything . [That is, anything I thought was important.]

All: All I gotta do is ... [A synonym for "just."]



Eventually I noticed that both Penny and Jeff were yawning. I suddenly realized there were many ways to put people to sleep with words, so I stopped talking and they both woke up.

Later, I reflected on the deep lesson underlying all these lullaby words. In effect, they are designed to discourage feedback by putting both the speaker's and the listener's minds to sleep. And no feedback means that the meaning of the statement containing a lullaby word cannot be clarified. And, if it's not clarified, it can mean almost anything— and that's always the beginning of trouble. If you want to avoid such trouble, start converting those lullaby words to alarm words—waking you up to potential misunderstanding, rather than lulling you to sleep. Just do it!

Constant Comments

Ultimately, the most dangerous lock language comes not from clients, but from myself. In some ways, the most powerful form of lock language is a statement of the form:

I am a consultant.

The forms of "be" are what I call "constant comments." (With apologies to your tea drinkers.)

If you say you ARE a consultant, then to me it sounds as if you're locked into consulting forever. To see the difference, compare these pairs of statements:

- a. I am a consultant.
- b. I am currently working as a consultant.
- a. I am a specialist in operating systems.
- b. One of my current special areas is operating systems.
- a. I don't know how to handle such an unruly client.
- b. Up until now, I haven't learned to handle such an unruly client.

Notice how the (b) statements all provide for the possibility that you can open some new door with your Golden Key, while the (a) statements—all constant comments— make you look and feel locked into one place forever.

My favorite Golden Key trick for unlocking these constant comments was inspired by Virginia Satir, but dedicated to the Nebraska farmer who was being interviewed by a reporter on the occasion of his one-hundredth birthday.

"Have you lived all your life in Nebraska," the reporter asked.

The farmer thought for a moment and then replied, "Well, yes ... up until now."

So, whenever you hear static language that seems to lock you or one of your clients into an unchangeable state, simply add, "... up until now."

Then watch the locks spring open.



[Back To Index](#)



Biography

Gerald Marvin (Jerry) Weinberg is an American computer scientist, author and teacher of the psychology and anthropology of computer software development.



For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including *The Psychology of Computer Programming*. His books cover all phases of the software life-cycle. They include *Exploring Requirements*, *Rethinking Systems Analysis and Design*, *The Handbook of Walkthroughs*, *Design*.

In 1993 he was the Winner of The **J.-D. Warnier Prize for Excellence** in Information Sciences, the 2000 Winner of **The Stevens Award** for Contributions to Software Engineering, and the 2010 **Software Test Professionals first annual Luminary Award**.

To know more about Gerald and his work, please visit his Official Website [here](#).

Gerald can be reached at hardpretzel@earthlink.net or on twitter [@JerryWeinberg](#)

More Secrets of Consulting is another book by Jerry after his world famous book **Secrets of Consulting**.

This book throws light on many aspects, ways and tools that consultant needs.

“Ultimately, what you will discover as you read this book is that the tools to use are an exceptionally well tuned common sense, a focus on street smarts, a little bit of technical knowledge, and a whole lot of discernment”, says **Mr. Michael Larsen**.

More Secrets is definitely useful not only to consultants but to anyone for building up his/her own character by implementation of the tools mentioned in day to day life.

Its sample can be read online [here](#).

To know more about Jerry's writing on software please click [here](#).

MORE SECRETS OF CONSULTING



Gerald M. Weinberg

TTWT Rating: ★★★★★

A photograph of a green, teardrop-shaped pendulum bob hanging from a thin wire. The bob is positioned over a surface of light-colored sand. In the sand, directly beneath the bob, is a circular pattern of concentric, slightly raised ridges, resembling a ripple in water or a sand mandala. The entire scene is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk

Beware The Hero !

By Elisabeth Hendrickson



In my paper-based WordCount simulation, a team creates a word counting system on index cards. Each participant chooses a role for themselves: Product Manager, Tester, Developer, Computer (channeling their inner robot), or Interoffice Mail Courier. I am the facilitator, and I play the role of "Customer." Each successful delivery generates (fictional) revenue. But as with real projects, producing a working product on a tight schedule can be challenging.

The team in the WordCount simulation was floundering. We were midway through the third round and it looked to me like the team wasn't even close to shipping.

Most teams are able to produce a basic system that I'll accept in my role as the Customer in the third round, or early in the fourth. That's important to me; teams need to enjoy at least a modest success. Total failure is depressing, and I have found that teams that fail are more likely to point fingers than glean insights about Agile Testing.

But sometimes, like in this case, a team will struggle to ship anything at all. So I have a variety of techniques for increasing the probability that the team will be able to produce an acceptable system by the end of the fourth round. Sometimes I give the team a little extra time in a round. Sometimes I ease up on my more persnickety requirements. Sometimes I push for specific practices that I think that they're missing during the mini-retrospectives between rounds. And sometimes, particularly if I sense that a little individual coaching would help, I gently and quietly intervene during the simulation rounds.

In this case, I had noticed that every time the Developers installed new code on the Computer, it caused some kind of problem. The most recent update caused the Computer to throw purple catastrophic error cards with every input. Worse, the code updates were coming very slowly, so I didn't think the Developers would be able to fix all the problems before time ran out.

I read the code deployed on the Computer, and immediately understood why the Computer was throwing catastrophic errors: there was a simple error in logic. I could also see why the Developers had not detected the problem: the error in logic was hidden in instructions that were so overly complex they were nearly incomprehensible.

So I decided that a Developer intervention was in order.

I walked over to where the Developers were working. One Developer was standing up, a little away from the group, writing code on green cards taped to the wall. The other five Developers were sitting around the table, talking and sketching ideas on notepads. I walked up to the guy standing up, working on green code cards. He was, as far as I could tell, the only one actually writing code.

"How's it going, Fred*?" I asked.

"Rough," he grumbled. "I have all these bugs I'm fixing!" He waved his hands at the collection of red bug cards taped to the wall next to the code. "And I'm also having to talk to the Product Managers to understand the real requirements, and also tell the Testers what to test."

"OK, so you're overloaded," I said, reflecting Fred's frustration back to him. "Are you the only one writing code? What are they doing?" I asked, gesturing to the other Developers.

The other Developers looked up at us. Fred replied, "I'm doing the bug fixes on this release. They're working on the next release."

"Yeah," said a Developer, holding up his notepad. "We're figuring out how to do the Sorting requirement."

"So your fictional company hasn't shipped anything yet, and you have one guy working on making the current release shippable and five people theorizing about how the next set of features could maybe, perhaps, be implemented?" I summarized the current state of affairs.

They all nodded.

"How about it, if someone helps Fred with the current release?" I asked. "Maybe it would help if someone paired with him on the code?" I looked to the other five Developers for a volunteer.

Fred looked offended. "No," he said. "That's not necessary. Let them work on the next release. I'm fixing the bugs. I'm almost done."

At first, I thought the problem was that the other five Developers were having too much fun theorizing instead of coding, leaving Fred to do the dirty work. But I suddenly realized that there was another dynamic at play.

Fred was enjoying being the go-to guy for this release. Yes, he complained that he had a rough job. But that complaining was his way of saying, "See how hard I work? See how much I can take on? I am carrying this whole release on my shoulders, by myself. I am a Super Star!"

I was not going to be able to talk Fred into accepting help. So I decided to leave Fred to his work, and check in with the Testers. I'd noticed that some of the "bugs" that Fred was supposed to fix weren't really bugs from the Customer's point of view, and I wanted to find out how the Testers were designing their test cases.

After checking in with the Testers, I checked in with the Product Managers. Now I was sure that all the requirements and tests were in sync and represented the real Customer requirements. I returned to Fred. He was plugging away with code fixes, batching them up for one big bang install.

"How's it going?" I asked.

"Good, good. Almost there!" Fred seemed frantic but cheerful.

I worried that Fred was about to install a lot of changes all at once. Installing one change at a time would have given Fred much better feedback.

Then I noticed a new bug on the wall, one with a purple catastrophic error card attached. In his harried state, Fred was ignoring incoming bug reports and test results. "What's this?" I asked, pointing to the bug.

Fred frowned and shook his head as he read the bug report. He tore it off the wall and rushed over to the Computer table. I followed. Fred slapped the purple card in the middle of the table. "What's this?" he demanded.

"An error," replied one of the Computers. "Because of this instruction," she pointed to a green card. "It doesn't make sense."

Fred leaned over, reading. "Of course that makes sense," he declared as he pointed emphatically at the offending green code card, "That instruction means that you ignore single quotes unless it's an apostrophe!" Fred walked through the instruction set with the Computer again, step by step, explaining what the code meant. Finally, satisfied that the Computer understood his intent, he went back to his coding wall.

After Fred left the Computer, we ran a Customer acceptance test. I fed the Customer acceptance test phrases as input to the Computer. They were a little closer to having an acceptable system: two of my four acceptance test cases were passing. Closer, but not there yet.

Time was up for the round. So I called "Stop!" and we went into our short debrief and mini-retrospective. The team was grim. The tang of failure hung in the air and made everyone a little edgy.

For half an hour, we talked about bugs and acceptance tests and requirements and alignment and feedback. The team decided to make some changes to improve their practices, and we began again. I felt confident that the team had the information they needed, and much more effective team practices in place, and that they would have an acceptable system early in the fourth round.

However, as we got into the fourth round, I had the sinking feeling that they might actually still fail.

Most of the organization was doing well, as I thought they would. They were working together effectively, collaborating to ensure that tests were automated, that the test results were visible, that requirements and tests were aligned, and that the bugs and new requirements were appropriately prioritized. The team as a whole was on track.

But Fred was all over the place. He was still working on those bugs. He was coding, he was talking to Testers, he was talking to Product Managers, he was arguing with the Computer about how to interpret his instructions. He raced around the room, frantically busy. And Fred's fixes were the only thing standing between the team and success. No matter how well the rest of the team did, Fred's inability to get those few remaining bugs fixed was dooming them.

It was not because Fred was incompetent. Far from it. Fred was very capable. Yes, the code he wrote was a little convoluted. But that wasn't the real problem. The real problem was that because Fred insisted on doing all the fixes for the first release himself, the entire capacity of the team was throttled

down to what a single person could do. And that one person was so overloaded he couldn't process the information he was getting—the test results and bug reports—that he desperately needed in order to make the code work correctly.

By taking so much on his own shoulders, Fred was doing something Ken Schwaber cautioned me about when I took the Certified Scrum Master class: he was being responsible to the point of irresponsibility.

Eventually the team did succeed. 16 minutes into the 15 minute round, the team shipped acceptable software and recognized revenue, and there was much rejoicing.

During the end-of-simulation debrief, I offered my observations about how everything landed on Fred's shoulders, and how that caused some of the pain they experienced. But I chose not to offer my opinion that Fred relished his role as The Guy Who Did Everything. I don't believe in public humiliation as a teaching mechanism.

I did, however, want to take Fred aside for a private discussion about his role in the team's near-failure. Unfortunately I didn't have a chance before the end of class. It's entirely possible that Fred still does not realize that he almost single-handedly brought down the whole company. In his mind, he's the guy who made the software work.

Fred's experience taught me a crucial lesson: **Beware the Hero.**

The Hero mindset is deeply ingrained in the software industry culture. As an industry, we've romanticized the image of the lone programmer hyped up on caffeine and pizza pulling all nighters, performing technological miracles.

Fred wrote the vast majority of the final shipping code. He fixed the vast majority of the bugs. He coaxed the Product Managers into clarifying the requirements. He helped the Testers know what to test. By all possible measures, Fred was a super star, a Hero.

And yet the team only just barely managed to ship as the fourth round went into overtime. It was an unnecessary near-miss: I've seen numerous teams run this simulation, and given the practices this team had in place by the end of the fourth round, they were well-positioned to crank out feature after feature and rack up a tidy sum in fictional revenue. The only thing that held the team up was the bug fixes that Fred had been working on, by himself, for half the simulation.

The team very nearly failed.

All because, Fred insisted on being the Hero.

** Not his real name.*

Discuss this topic on Quality Testing
[Click HERE](#)



Biography



Elisabeth Hendrickson is the founder and president of Quality Tree Software, Inc., a consulting and training company dedicated to helping software teams deliver working solutions consistently and sustainably.

She also created Entaggle, a community-based site for giving and getting professional recognition. And she founded Agilistrystudio, a practice space for Agile software development in Pleasanton, CA. A software professional for over 20 years, Elisabeth has been a member of the Agile community since 2003. She served on the board of directors of the Agile Alliance from 2006 - 2007 and is one of the co-organizers of the Agile Alliance Functional Testing Tools program. Elisabeth splits her time between teaching, speaking, writing, programming, and working on Agile teams with test-infected programmers who value her obsession with testing.

You can find her on Twitter as @testobsessed.

[Back To Index](#)

Visualising Quality

By David Evans and Gojko Adzic

If quality falls and no one is around to notice it, does it make a sound?

- David Evans and Gojko Adzic

The famous question about trees falling in uninhabited forests, which makes us think about the difference between sensation and reality while contemplating global deforestation, is oddly applicable to many of our recent consulting engagements. Several large software teams in banks, accountancy firms and online services companies have called upon us to help with their quality problems, each team with a unique explanation for those problems, but all essentially stemming from the same root cause. They all suffer from quality mis-definition.

All these teams implicitly define quality through a lot of data, much of which is irrelevant or completely missing the things that are really important to quality. They all track bug counts, some also track story points and produce burn-down charts as indicators of their product and process quality. Gerald Weinberg [1] defines quality as "Value to some person". Yet none of these teams measured the value that their system provides, or to whom, or how that matched the expectations of important



categories of people. The most extreme case was an accountancy firm where we interviewed 15 stakeholders, asking them for the top three things the system should deliver, and got 30 different answers. There is no software team in the world that could satisfy such a diversity of quality expectations. In the forest of bug tracking information, if stakeholders' implied expectations are no longer met, the fall of quality truly makes no sound and there is no one around to hear it. But the quality falls nevertheless, and someone is up for an embarrassing surprise later when the software goes live.

A lack of clear expectations around quality often makes teams focus on testing software in proportion to the size or complexity of the underlying components. The more functionality or lines of code a component contains, the more testing attention it gets. While this may seem logical on the surface, it is unjustified and in many cases irresponsible, if it does not reflect the true balance of importance to stakeholders. We often find that a very small portion of the system carries the most valuable and important functionality and hence the bulk of the risk. Eric Evans calls this the "core domain" [2] and it is where most of the investment in quality assurance, improvement and measurement should be focused. Even if this core domain has relatively few lines of code or lower complexity, a small improvement in that area will pay off handsomely in business returns. Likewise, a small problem in that area is likely to have a significant impact on the users and stakeholders.

Why do so many teams continue to track and measure the wrong things, and fail to take account of stakeholder value and risk? Douglas Hubbard [3] explains both problems as a Measurement Inversion:

"The economic value of measuring a variable is usually inversely proportional to how much measurement attention it usually gets". As a community, we are generally guilty of measuring what is easy to measure, rather than measuring what actually matters. In particular, we have to move away from the idea that quality can be adequately expressed by numbers of defects, just because they are easy to count. We must take the time to understand and quantify things that are both measurable and also reasonably reflect the capability of a system to support the interests of its stakeholders.

We recently helped a large media company understand the causes of the perception of poor quality of their software. Their back-office users often complained about problems with unreliable article publishing queues. Editors would often publish the same article several times, because it did not appear when they expected it and they did not know if it just got stuck somewhere or if there was a problem with it. Once we understood that this aspect of the system was important, the team bought a monitor to show queue status and queue processing time for the most recently published article. This helped business users see what to expect when they publish an article and also alerted developers to potential problems. Furthermore, it stopped complaints about quality of that component. **The moral of this story is that a very important aspect of the role of QA people is visualising the right kind of information to stakeholders, so that they can make informed choices.**

James Whittaker has spoken at several conferences recently about the Attribute-Component-Capability matrix they use at Google [4]. This matrix charts out the relationship between three things:

1. What is important from a business perspective (attributes)

2. System capabilities that support these attributes

3. Software components that are involved in providing these capabilities

Google testers use the matrix to visualise test coverage of important risk areas and to focus their efforts. Our experience with trying this approach is that we often got bogged down with the complexity of representing multiple stakeholders, which the matrix doesn't cover directly. On the other hand, the technique of creating an Effect Map [5] does cover the multiple-stakeholder aspects of a system very well, albeit for planning a future piece of software, not for something already delivered.

We combined the two approaches in what we are currently calling a Value Map - a visualisation of the quality landscape for a software product. We use it to define and direct testing efforts for large and complex software systems. The goal of a value map is to create a shared understanding of expected quality for a software system at a semantic level. It helps us define what needs to be measured and how to inform decisions of business stakeholders regarding product releases, as well as process and product improvement. It also facilitates a meaningful prioritisation of testing work and investment in testing, enabling teams to focus on areas of software in proportion to risk and importance, not necessarily lines of code or technical complexity.

We create a Value Map by drawing a mind-map that starts with the system and expands one level at a time by answering the following questions:

1. Who uses our system, and why? Who are the stakeholders and which goals does the system help them achieve? The stakeholders become the second level of the map and the stakeholder goals or activities are the third level. One example for an investment fund might be that fund managers use the system to monitor their profit-and-loss performance.

2. Assess risk and importance of these activities, either informally (at Google they use voting) or using a technique such as the Kano model [6]. This will help to define both the prioritisation of future testing work but also the prioritisation of expanding the map. Regardless of the technique, it is important to get a rough ranking of activities compared to other activities.

3. For each activity, what do the relevant stakeholders expect from the system? What do they assume about how well the system will support their activities? This effectively lists expected product qualities or characteristics and defines what needs to be measured. In the Attribute-Component-Capability matrix these are the Attributes. Continuing with the financial fund example, the managers might expect correctness and completeness of the data. They might also assume that the reports for a previous day will be ready the next morning (timeliness).

4. For each product quality, what are the key scenarios that could be used to prove or disprove that the relevant activity is supported, or the degree of how well it is supported? This defines how to measure stakeholder confidence. In the financial fund example, a scenario to prove completeness might be booking a transaction against all relevant funds and checking that they all appear; key scenarios for correctness would include booking new transactions, amending transactions and canceling transactions over different day boundaries and validating the aggregated results. For timeliness, key scenarios might be running the four biggest reports.

5. For each scenario, what would represent "good enough" from the point of view of a stakeholder? This defines how to interpret the measurements. For many scenarios this will be a range. A structured way to do this could be to use the QUPER (Quality Performance) model [7]. For the previously mentioned scenarios, timeliness might be good enough if four biggest reports finish in less than 3 hours. For correctness, good enough might be matching the expected aggregation 100%.

6. For each scenario, what are the software features and components involved in these activities? This defines the system under test.

Mapping all this information creates a visual landscape that covers stakeholders, activities that need to be supported, and their expectations of the quality characteristics of the system. This defines quality at a much better level than the presence or absence of bugs in unqualified areas of a system that are all treated equally. Defining how and what to measure to prove the expectations or mitigate the risks; defines what and how to test or monitor. Combined with the assessment of risk and value on the level of activities, this clears up matters of what is really critical to do, what is nice to have and what areas of the system we should focus on only when there is absolutely nothing more valuable to do.

All this information already exists in people's heads but it isn't made explicit or available to the entire team and the stakeholders. Mapping it out visually goes a long way to facilitate a meaningful discussion on priorities, scope and what constitutes "good enough" and to create a shared understanding of that among the team and its stakeholders.

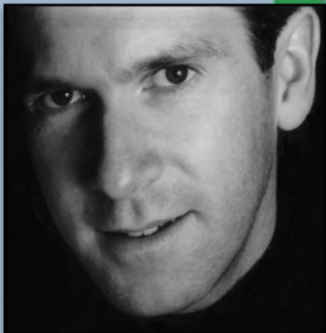
There is a bit more to this technique to deal with very complex systems or activities that need to be further broken down, but this is the gist of it. We are still refining this technique but so far the results are very promising. Try it out - we'd love your feedback!

References:

- [1] Gerald M. Weinberg, Quality Software Management, Volume 1. Dorset House, 1991, ISBN 9780932633224
- [2] Eric Evans, Domain Driven Design - tackling the complexity in the heart of software, ISBN 9780321125217
- [3] Douglas Hubbard, How to Measure Anything, ISBN 978-0470539392
- [4] Watch the video <http://blog.utest.com/more-bang-for-your-testing-buck-follow-up-qa-with-james-whittaker/2010/12/> especially after 27th minute, and see the slides at <https://docs.google.com/leaf?id=0B4fT-BFGDnQkMDE5OTk0ZmUtODQ4Ni00NTM3LTlkMGEtNjg1MTdiMzM5Nzg5>
- [5] <http://gojko.net/effect-map>
- [6] <http://www.betterproductdesign.net/tools/definition/kano.html>
- [7] See <http://oredev.org/videos/supporting-roadmapping-of-quality-requirements> and <http://gojko.net/2009/11/04/quper-model-for-better-requirements/>

**Got to say something? Discuss this topic on Quality Testing.
Click [HERE](#)**

Biography



David Evans consults and coaches agile teams to help them improve the ways in which they deliver quality software.

Get in touch at david@thinkalikeconsulting.com



Gojko Adzic is a strategic consultant who helps ambitious teams, from investment banks to web portals, to improve the quality of their software products and processes.

Get in touch on <http://gojko.net> or [@gojkoadzic](#)

[Back To Index](#)





Software Testing & the art of staying in Present

By Anuj Magazine

When I first wrote an article on **Soft skills that make a tester** some years back, little did I capture the importance that power of concentration plays in our profession. As I grew more in the professional life, I am beginning to get more convinced that most of the inefficiencies while testing or even while dealing with regular work related stuff stems from the notorious ability of our mind to wander at will. While it's quite bold of me to relate lack of concentration as one of the major reasons for quite a lot of inadequacies at the work place but a little introspection would reveal that lapses in concentration during the day actually keep us away from being at our best.

Consider following situations-

- A tester working to perform Exploratory testing in a well-defined session often sees himself lost and with his mind miles away from task at hand. Does a 1 hour session mean that a tester get to spend entire one hour on testing? Probably No even if a tester completely isolates himself/herself from the distractions around his work.
- How many times does it happen that while performing one task you are either thinking of what happened before the current task or what is going to happen after the current one?

A Voice on "Teach-Testing" Campaign!

I strongly agree to this.

I am a quality engineer and in colleges all I learned was how to develop an application (any). I never knew what quality is. So it took time for me to understand the importance and role of a Quality engineer.

Testing an application is equally important as developing an application.

- Arvind Verma

I come from a BS in Computer Engineering Background. A lot of engineering disciplines require a two part Senior design Class. The first semester is spent designing, and considering much of what you would build, and the second typically spent building it. While testing might be mentioned at some point in the process of the 'block diagram' approach, let's be honest, it's often an after thought, or given only cursory glance. I think a Course on Software testing could be huge for the industry, but I also feel that if left on its own, won't fix the problem. Many Software classes, be they Computer Engineering, Computer Science, or Software Engineering spend so much time on making things 'work' that they spend little time on verifying 'what works'. Multiple courses need a testing element added into them, even if it's only 1 point on a 10 point quiz. The question is how to convince Academia to spend time on this issue, because it is important.

I whole heartedly support this effort. I think many people think at least one, perhaps two if an advanced course on Software Testing would be of great boost to those who end up in the field. I see immense value in this in curricula for Software Professionals; however, it could also be a double edged sword. My comment is mainly directed at the idea that giving more than a single course devoted to testing may not be enough to draw people in.

- Tim Western

I agree Software testing should be taught to students, there should be some syllabui in whcih students will be taught both Manual as well as functional. There should be choice whether they want to become developer or tester. As many of the students still don't know that IT has testing as different field.

- Gagan Talwar

- While working on something important, you tend to get distracted with an email arrival and tend to spend time there even though it may be as trivial as a joke from a friend.

- If you actually got to use a tool called **Session Tester** meant for assisting testers in performing Session based testing, it has an interesting feature called "Prime Me". On clicking this button, a tester is given some useful suggestions asking him/her to focus on the task at hand. Some suggestions such as "Try touring in different ways", "Try something radical", "What do you see that you didn't see before?", "What would Grandma do?" etc. This feature is actually quite handy to bring the tester's mind back to where it should be. Why do you think such a feature would be needed?

I feel that looking back at your day or even last one hour, you would quite appreciate the fact that our mind tends to be one of the most dynamic entities. At the same time, there is also a growing realization that at the foundation level of every success, there is an individual's power to focus on the goal till it's completed successfully. In this context, the greater challenge is how can one tame or control one's mind to get the best out of it at will.

What is concentration?

The Oxford dictionary definition of the word "concentration" is "the act or power of focusing one's attention".

One of the best definition of the word "concentration" comes from **Geet Sethi** (7 time World Billiards Championship holder) when he says, "*Concentration is simply remaining in the present. The longer you can remain in the present, the greater will be your span of concentration.*"

This definition quite beautifully sums up what concentration is all about.

Is it easy to concentrate and focus on something?

I think **Geet Sethi's** definition of concentration makes it seem quite simple. In reality, is it really simple? I bet you would agree with me in answering this question as "No". **Sachin Tendulkar**, arguably the greatest Batsman in the sport of Cricket, in one of the interviews to Wisden stated, "*The toughest thing about batting is to clear your mind. The mind always wants to be in the past or in the future, it rarely wants to be in the present.*"

My best batting comes when my mind is in the present but it doesn't happen naturally, you have to take yourself there. I am not able to get into that zone as often as I would like but, when you are there, you don't see anything except bat and the ball."

Don't you think, Sachin's above statement proves that it is so human to have a mind cluttered with thoughts that either take you months ahead of current time or may be years behind? This is a constant battle that every person faces irrespective of his/her stature and greatness is actually bestowed upon the people who are more consistently able to tame their mind to move in the direction they have set for their lives.

That is what Sachin has referred as being in "Zone".



As another cricketer- **Aakash Chopra** recently put it in his recent articles on the same topic- *"The mind has the peculiar ability of wandering off at the first available moment, and it doesn't need any permission."*

The Power of focus in Software Testing:

For those who have experienced the job of Software testing, would agree with me when I say that it's an intense job. To do the testing perfectly and to get the desired results requires a tester to have more than just the Technical skills required to do the job. Consider a scenario in which 2 testers with similar educational background join an organization and undergo similar training but while at work one tester gives absolutely wonderful results while the other remains average. There can be multiple factors leading to this situation but one major factor leading to greater performance of an individual has been the Power of focus or concentration that one exhibits while working on a task. For people who are passionate about Software testing, would find their attention span on task at hand i.e. testing the software more than the average testers.

Correlating that with Sachin's statement in my previous point - when a tester is in **zone** he/she always sees only the Software to be tested and works with it with full attention, eliminating all other distractions.

The more a skilled tester reaches such a zone, the better he/she will get at the craft of Software testing.

I would call this as **"Zone of Accomplishment"**.

Is it possible for a tester to get in "Zone of Accomplishment" every time he/she tests?

業
積

I won't be forthcoming here and say that the answer to this is "Yes" just because we are humans and cannot always be perfect but one thing is true for a fact that the great testers get into this zone more often than others. In my experience, I have realized that often we try and fight with the thoughts in our mind to focus on task at hand. A mind in current state may have zillion thoughts such as any unfinished work, thoughts about the next day, thoughts about your personal life, thoughts about traffic, weather etc.

One of my realization have been that instead of fighting with mind to get it to desired state- as a first step acknowledge that its natural for mind to wander and accept the

status quo. Don't really be hard on yourself, let the mind work with multitude of thoughts while you carve for maximum attention on the task you want to focus on.

Another idea I have found useful is to defocus and allow your mind to wander and then focus back again. The more we allow mind to work naturally, the more attention spans we would be able to get back in return.

I would really like to hear from you on your experiences towards staying in present and getting in to "Zone of Accomplishment" while testing.

Do share your thoughts.

Credits and inspiration:

The Inspiration of this article comes from a beautiful article written by [Aakash Chopra](#) . You can find the article [here](#) .

Biography



Anuj Magazine is a Software Testing practitioner currently working at Citrix R&D India Pvt. Ltd. He likes exploring new avenues and their intersection with Software testing profession such as Handwriting Analysis, Assembly line approach, Six thinking hats etc. He is passionate about Software testing and has been a regular speaker at conferences, such as QAI, STEPIn forum, SoftTec etc. and has spoken on several software testing topics.

He has written articles in www.stickyminds.com , TheSmartTechie Magazine, and writes frequently in his blog Creative Tester .

Anuj also holds a professional qualification in Handwriting Analysis. He can be reached at amagazine@gmail.com & at twitter [@anujmagazine](https://twitter.com/anujmagazine)

[Back To Index](#)



No Problemo ! We will be publishing your Feedback/Comments and even Answers to the Questions that you have for the Author on his/her article that gets published in our magazine.

Do write us with your feedback
send it to teatimewithtesters@gmail.com :

- Your Name
- Brief Introduction about Yourself
- Article Name
- Your Feedback/Questions

➤ Your Feedback, Question, or Comment

Make sure to write **Feedback for <Article Name>** in your Subject Line.



A photograph of several young students in a classroom, seen from behind, with their hands raised in the air. They are facing a chalkboard that has some faint writing on it. The students are wearing colorful shirts: light blue, red, orange, and green. The entire image is framed by a thick black border.

In the school of Testing

for your better learning & sharing experience

Finding your niche in Software Testing

By Anne-Marie Charrett

Many testers feel uncertain about career path they should take. There are many choices. Testers can specialise in the types of testing performed, such as automation or exploratory testing. Or perhaps the goal is to become a test manager or to specialise in one area of testing, becoming an expert in that field.

Hey, maybe you feel the life of a consultant is for you!

These are all worthy careers and many testers find it hard to decide what area they wish to work in. There's a perception that once you decide on one career path, it's hard to shift around. This is driven by lack of knowledge and I think to some extent a fear of the unknown. That's understandable; in some ways these choices appear to be big decisions with large implications. But personally, I think as testers its worth trying many different areas out before deciding on one career path. After all, until you test it for yourself its all conjecture.

By trying many different areas, you can then evaluate for yourself how you feel about a particular topic. How did I enjoy this area of testing? What did I like about it? Does it fit with my general aspirations, my values?

I continuously evaluate where I am in my career, evaluating what I have learnt and modifying my choices based on my experience. I do this because no-one else but me, really knows what's best for me. Others advising me often appear to have my interest at heart. Too often though, it turns out



there is some other motivation such as the company’s interest lying at the centre of this change.

I’m also wary of looking within the testing community for advice on the best career path. Here’s an example of software testing career path I recently saw in a brochure.

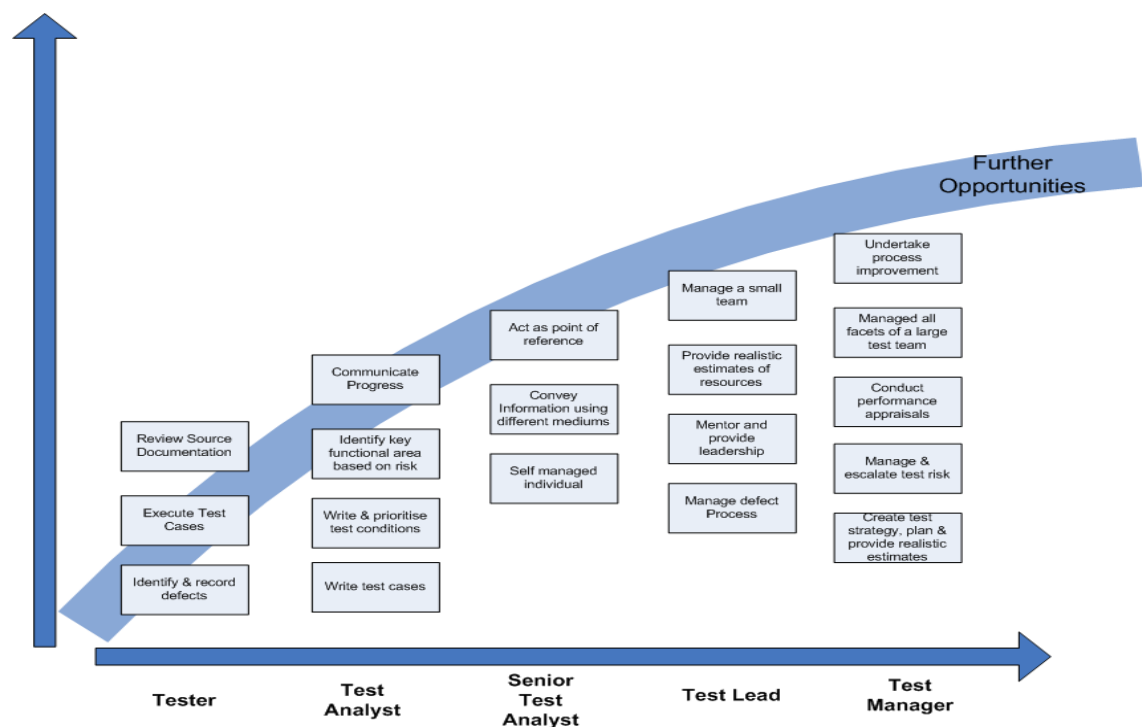


FIGURE 1 - SOFTWARE TESTING CAREER DEVELOPMENT MODELⁱ

Typically, I would find this type of advice amusing, until I realised that people actually take this stuff seriously. Few testers I know of, have followed a career path such as this.

As an exercise I decided to plot my career path to date and see what it looked like. You can see it below.

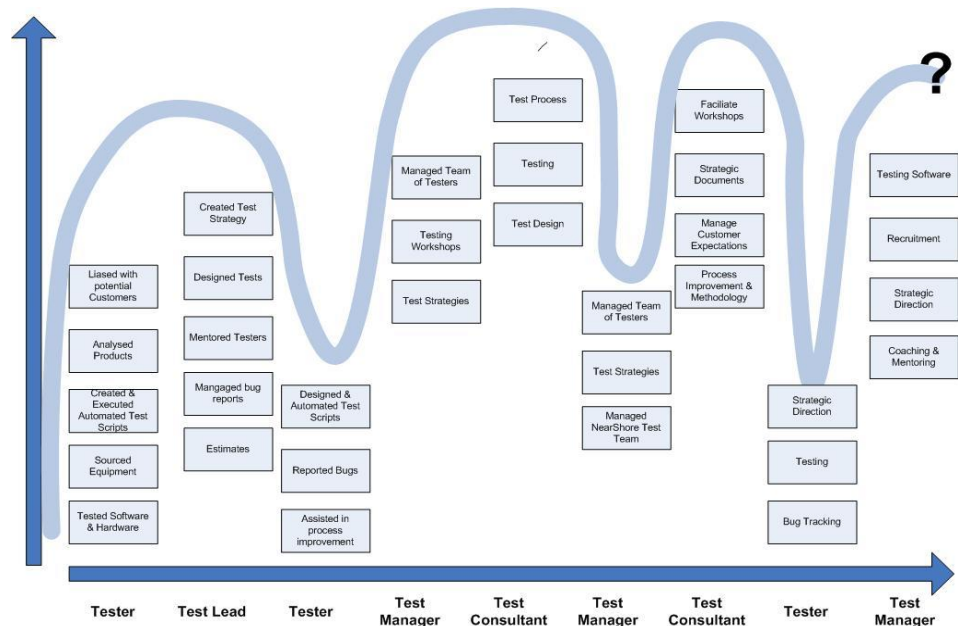


FIGURE 2 - PERSONAL SOFTWARE CAREER DEVELOPMENT MODEL

It's nothing like the typical career path! In fact, I found it impossible to tie in my experience of testing with any of the above described roles. The whole idea that a career may be mapped in such a way is too simplistic. Imagine if I asked you to write out your resume for the next ten years, stating the roles and tasks you will perform. I would like to think, intelligent reader that you would refuse. You would tell me its "impossible" to predict such a path. Yet when we try to plan out our careers in such detail, this is exactly what we are doing.

Most 'testers' fall into testing and then discover a passion for it, they don't plan it. But when looking around for career options, they're shown the 'successful' career path as described in the first diagram. You start as a tester and then 'progress' to Test Analyst, Senior Test Analyst. If you are seen to have any skill or passion, you get to climb further up the corporate ladder to Test Lead until finally you reach the ultimate pinnacle of a tester's career, the test manager.

But why is management the top of a tester's career? Why is this position elevated and valued so highly? What makes management and communication so well regarded above other testing skills? Granted, these skills are important and it's right to value them, but why over other essential testing skills? Tester skills are diverse and broad – more so these days than any other, so why do we hold onto this obsolete concept of a single career path? What if your project management skills are poor, does that mean you will never "succeed" in testing? It's ludicrous!

A skilled tester with many years of testing experience is a highly sought after asset, any recruitment agent will tell you that! A skilled tester that is passionate and experienced and technical in my opinion is worth 5 test managers who have forgotten the beauty of testing and have hidden their heads in metrics and project plans.

A good test manager is an awesome individual. They create space for testers to perform their art; they remove obstacles before the testers are even aware they exist. Good Test Managers feed and communicate risk to the rest of the organisation. They also challenge their team to grow and improve their testing craft.

Don't be fooled into believing that being a test manager is the ultimate goal. There are plenty of testers who have fallen into this trap, flown too high to the sun, only to discover that this is not where they wanted to be. If they're lucky, they get back into testing, but this can be hard decision to make as it often requires a salary sacrifice.

And anyway, excelling as a good tester; does not automatically mean that you will be a good test manager. Even if you did have the necessary skills, this path may not be attractive to you. Often passionate testers are coerced into becoming test managers because they have drive and skill. I think this is a real shame.

As a tester, your career path is your responsibility, not your manager or your company. You need to work out what you value, what motivates you. It can be any number of things. For example, it could be wealth, or recognition as an expert, or perhaps it involves leading and mentoring people. Perhaps it's to become the best in your craft.

Ask yourself what aspects of testing do you enjoy and then figure out how to incorporate those in your career path. But be reasonable. A career path is only one possible route to your goal and as you are probably aware, life often fails to work the way you planned it. It's impossible to predict exactly how things will turn out. Be flexible and review your plan frequently.

Most importantly, be open to exploring your career path. Be open to opportunities and be open to change. There are many reasons why I've chosen one job over the other. In some points in my life it's because I needed the money, other times it's because I wanted to learn a particular skill. Whatever the reason, own your choice and give it your best shot.

One reason why I chose a career in consulting was that I found it hard to work for other people. I wanted to be my own boss. I and my family have counted this cost many times. I've come close to considering the benefits of a pay-check (trust me there are lots!) but fundamentally this is where I want to be. I worked on my own definition of success and I've moulded my career around that.

Can you say the same about your path?

Taking responsibility for your own career and how it flows and giving yourself permission to try things out is what life is all about. It's a learning experience and it helps to treat it as such.

Enjoy, and let the exploring begin.

¹ This development model was based on information sourced from Plan it Marketing Brochure.

Got Something to Say?

➡ Discuss this topic on **Quality Testing**.



Click bellow to jump in



Our Community Partner



Biography



Anne-Marie Charrett is software testing coach, trainer and consultant and works on helping individuals and organisations improve their testing skills.

She will be delivering a workshop on Career Management for Software Testers at CAST 2011.

An electronic engineer by trade, testing chose her, when in 1990 she started conformance testing and was hooked. She has been testing ever since then.

She is passionate about Exploratory Testing, motivating testers to improve their testing skills and renewing their passion for testing.

Anne-Marie offers free IM Coaching to testers and developers on Skype (id charretts)

She also assists in e-training on the BBST Foundation Course run by the Association for Software Testing

You can find out more about Anne-Marie on her blog, <http://mavericktester.com> or through her company <http://testingtimes.com.au>

Back To Index





in Mobile Testing !

By Karen Johnson

I find software defects by asking questions. I ask myself questions while I'm testing and those internal questions become test conditions. I ask developers questions throughout the software development process; sometimes *just by asking a question*, defects are discovered. I have also found that asking one question frequently spawns a series of questions, all of which helps me generate testing ideas. Questions – asking questions is often like software defects – questions tend to cluster in collections just as software defects tend to nuzzle into nests.

When I poke around awhile in a specific area of a software application, I can usually come up with a series of questions and eventually uncover a defect. Getting my mind to think in questions, especially “what if” questions gets my testing mind moving.

In the mobile arena, it seems the market is currently in the mindset of *proving* mobile works but it's time to push past demonstrating mobile coolness and applying a tester's mentality of asking questions as we push to find issues lurking in our products. Thinking in questions helps find software defects, so here's a series of questions grouped in topic clusters to get you asking questions and finding mobile defects.

Network connectivity: Is there a network connection? If there is no connection available and the user is attempting to login or access an app or website, checking to see how login handles this condition is one checkpoint. But a user could already be logged in and then lose connection at any point in their session, so the condition of no connectivity or loss of connectivity can present many tests to see how gracefully your mobile app or website handles no connection.

What if the connection is fading or intermittent? I can almost hear a developer's voice in my head saying: how would that ever happen? But if we think about a mobile session a user they may easily be traveling and connectivity may go in and out – not just be on or off, working or not working but instead be intermittent. After all, I use my mobile device on trains all the time and as trains go in and out of tunnels, I've experienced intermittent connections. In the "olden days" of application testing, I would pull and replace a network connection cable to readily simulate this experience but now with mobile testing this is not as easy of a condition to replicate and yet a scenario that is *more* likely to occur than a user seated at a PC.

Session Interruption and Timeouts: When testing on a mobile device, think about the user experience and the fact that the very nature of mobile devices is that users are on the go – this changes our basic construct of what a user session is like. Imagine I'm using an application and receive a phone call? Does my app session timeout? Will I lose connection to my session because my session is disrupted by using some other feature on my mobile device? How long does my secure session stay open?

Batteries & Power Supplies: I have not directly experienced any issues with low or dying batteries, but I did hear from a tester that they had an issue with an application when the battery was dwindling. Like an intermittent network connection, the session was sluggish and they experienced connection issues. I wonder what would happen if I was testing on a phone where the battery was low and I plugged in the device. Would a surge or change in power supply impact my app session?

In a situation where testing is taking place with vendors like Device Anywhere or Perfecto Mobile (vendors who offer dozens if not hundreds of handset device to test with) these scenarios are difficult if not impossible to simulate since their devices are plugged in and always charged. This is where some manual testing with a few physically available devices is helpful, you can experiment in ways with a physical device that you cannot with an emulator or a mobile testing vendor.

Eating The Dog Food

Some years ago, I had a boss who used to say it was important for each of us to use the software we built for our own personal needs both in and outside of the office. He used to say it was case of us "eating the dog food." His point was that each of us was more likely to find flaws and gain ideas for product enhancements if we used the product for our own purposes. I've remembered that philosophy for a long time now.

This is one reason why I use my own phone frequently to see what my experiences are in a myriad of conditions, conditions that I may not experience in a test lab. Simply put, an office may be too pristine of an environment to test the reality of a mobile session. By using my own mobile device frequently I feel I'm gaining a better sense of conditions and situations that mobile users encounter. I'm also gradually developing a sense for "what to other companies do" in terms of their mobile apps and mobile websites.

As a personal experiment I've decided to use the Starbucks mobile app at least once a week for the next few weeks. (If you know anything about me, you know coffee is a significant part of my life.) In the last two weeks alone my experiences have varied in ways I'm sure I would not have encountered sitting in an office *pretending to be mobile*. Flying is another frequent activity of mine so I've been experimenting with American Airlines, Orbitz and Flight Track as I look up flights, track flights and travel.

I've been looking at other activities I regularly do and checking to find if there is an app and then using that app in as many real-life scenarios as I can. The way I look at my mobile usage is this: anything I do on a mobile device improves my own knowledge, experience and awareness. If you want to develop your own mobile testing skills, eat the dog food (or find an app that does.)



SD cards: Does your application store data on an SD card? What if the card is missing? What if the card is full? What if a different SD card is loaded into a device than what is expected? What if the card is switched during a session? I personally have not worked with an application (yet) that has stored data on an SD card but the possibility presents another set of conditions to think about. Does your app give a user an opportunity to store data on a card?

Bar codes: I often use Amazon's mobile app to look up book reviews and book prices when I'm in a bookstore. Sometimes the bar code scanner isn't able to pick up the code and I have to manually enter a book title. Does your application use a bar code? What happens, and how gracefully does your application function if a bar code cannot be read? I haven't tested an app that uses bar codes (yet) but I'm building my own experiences with different mobile apps and when/if I have a chance to test using bar codes, I'll have that background to draw upon.

Permissions: Most mobile apps require permissions to be granted in order to download and use the app. What if the user makes a change after downloading an app – you might wonder how a user can do this –so consider the following scenarios:

1. If an application needs location information and I accept the permission request, download, install and use the application but later shut off location detection – what happens?
2. If an application needs checkout or buying information and I accept this permission request, log into and use Google Checkout - download, install and then use the application but later log off from Google Checkout? What if I change my credit card information with Google Checkout what happens?

App or Website? "There's an app for that" – seems like a common expression these days – enough to lead you to believe that all companies have created a mobile app and that all testing on mobile these days is about mobile apps. But that's not the case. There are websites to test on mobile devices. Testing a website on a mobile device has a different set of challenges than mobile app testing. Consider a few "what if" scenarios for websites on a mobile device.

Mobile Browsers & Settings: To begin with, there's the factor of a mobile browser. Which browser? What if the user has downloaded another browser on their device and they're not using the "native" browser for the device? Do the browser settings impact the website you're testing?

Secure pages: Are there secure pages? Does the site notify the user when they access a secure page?

Redirects: Are there pages of the site that are not supposed to be accessible on a mobile device? What about redirects?

Navigation: Navigating on a PC is very different than navigating on a mobile device. Has the site been optimized for mobile? Can you navigate through lists? Links? Email pages like contact us?

Website Maintenance: Is the site down for maintenance? Is there a maintenance page designed to be shown on a mobile device? How's the display? Does the maintenance page have any links? Are the links accessible?

Thinking in questions, especially "what if" questions is only one way to approach testing but it is a good way to develop testing ideas as opposed to getting locked into a mindset that proves a mobile app (or website on a mobile device) works.



Our Community Partner



Your one stop place for Mobile Apps Testing



Pairing with an Expert

By Michael Larsen

One of the challenges being a lone gun tester is the fact that, often, you don't have someone else to talk with or ask questions. Sure, you can talk to developers about issues and areas you have concerns about, but that's not what I mean. It's rare that time will allow for a tester to consistently sit down with a developer and just ask broad and open-ended questions about a product, a technique or an idea. Larger test organizations with many testers allow testers to have this opportunity. Frequently, the Army of One tester ends up doing most of their thinking or brainstorming alone... but they don't have to.


Over the years, I have had some of my best testing experiences by stepping out of my day to day routine and asking someone from another part of the organization to sit down and test with me. Generally, the time commitment doesn't need to be a big one. Ninety minutes is often plenty of time, and the level of insights gained more times than not make the time very well spent. One of the more memorable experiences I had was when I was working at a company that made a software application dedicated to helping Immigration attorneys and companies that hire foreign national keep track of the many steps in the process. I had developed some knowledge of this area over the years that I worked there, but I would never consider myself an "expert" in this field. I did, however, have access to an Immigration Attorney who was working with our company to make sure that the legal foundation was solid. He asked me one day if I would be interested in sitting down with him for a pair testing session, with the idea of "asking the product some questions".

As we sat and tested, we discussed a number of aspects about the product that looked "interesting" (a euphemism for "something's not right"). Many of these situations consisted of items and pages and applications I'd looked at countless times, and felt they worked as designed. They did, but the domain expert can often see things in ways that a tester can't (at least at first, testers are notoriously fast learners). This is indeed helpful, but what I found more valuable was the chance to expand my own ideas as to how I could sit down and ask questions of a product, and more specifically, ask questions that users might ask themselves, even if many of the questions seemed far-fetched or unlikely

Through this single testing session, I discovered several areas where I felt that it certainly provided solid coverage. I realized after this session that I could do lots better. I also realized there were some areas that I could, at least for a little while, pretty much ignore (not completely, of course, but I was able to learn some areas that could certainly be put on a back burner while the more important areas received the proper coverage needed. but certainly give them less emphasis for a time). Having such a valuable resource on a regular basis can certainly be a blessing, but it's not practical much of the time (maybe most of the time). That's OK, because in this area, a little can go a long way. Asking many of the people to participate daily would be problematic, if not impossible, but I've found that, given an option of once a month or so, it's surprising at how many people are willing to sit down, roll up their sleeves and muck around for a half a day. Spread out among enough contributors (developers, support staff, marketing, sales, design, operations, etc.) you can get a lot of unique input to help energize testing efforts or, at the minimum, consider questions you may not have thought to ask yet.

While you may be lucky to meet someone who has all of the domain knowledge and some skill in testing as well, often the two do not come together in the same person (they can of course, but for many of these domain experts, testing is not one of the main things that they excel in). That's OK because that's what I am there for. By providing insights to testing, I spread some seeds as to the way that everyday users (and experts) can use the software in the way it was intended. Likewise, I can learn vital information as to how these domain experts think about the product, and how they use it. By combining a tester's skill with a domain expert's knowledge, the possibilities for testing and test improvement are endless.

Biography



Michael Larsen is a lone tester in San Francisco, California.

He has spent seventeen years in testing and test organizations, ranging from network routers and web caching devices to distributed databases dealing with the legal and entertainment industries.

Michael is a Brown Belt in the Miagi-do School of Software Testing, an instructor with the Association for Software Testing, and the co-founder and facilitator of Weekend Testing Americas.

Michael blogs at www.mkl-testhead.blogspot.com and can be found on twitter at @mkltesthead.

Please contact Albert Gareev (agidale.ag@gmail.com) or Michael to know more about Project Sherwood.

Back To Index



Why PROJECT SHERWOOD?

Because "in the cloud and by the crowd" is where we believe the future of test education and test improvement to be. We were inspired by tales of "Robin Hood". For example, according to Sir Walter Scott (See: [Ivanhoe](http://en.wikipedia.org/wiki/Ivanhoe) <http://en.wikipedia.org/wiki/Ivanhoe>), Robin Hood's band of "Merry Men" was largely made up of Yeomen.

The term "Yeoman" (Reference) nowadays suggests someone upright, sturdy, honest and trustworthy [...]. The term is used in contexts such as: "The forester provided 'yeoman service' in finding the lost children in the woods." "The security guard did 'yeoman's work' last night by staying alert and preventing a break-in entry after working very long hours in austere conditions."

Archery is an art, one that requires life-time practice and skill. So is testing. Longbow archers ruled the battlefields. They've proven to be effective in defense and offense, against both cheap hordes of militia and expensive heavy-armored knights. Yet yeoman archers needed only one simple 'thing' to be so: continuous practice, with life-time sharpening of their archery skills.

This is how we see our mission as practitioners of the skilled craft of exploratory testing. We view exploratory testing as a personal discipline. In light of this, we also want to practice working together and in concert with one another to become even more effective.

Sharpening our own skills and helping others to sharpen their testing skills is not the only mission.

We are aiming at bringing out and highlighting the results of testing products using session-based exploratory testing, as well as making people and businesses aware of these methods.

We are willing to establish a kind of "field certification" based on continuous professional growth, practical accomplishments, peer recognition, and a true contribution.

These, of course, require taking on testing projects and delivering visible results offering our services on a freelance basis.

Testing...



with Anurag

Getting Started with Mobile Apps Testing

Biography



Anurag Khode is a Passionate Mobile Application test engineer working for Mobile Apps Quality since more than 4 years.

He is the writer of the famous blog **Mobile Application Testing** and founder of dedicated mobile software testing community **Mobile QA Zone**.

His work in Mobile Application testing has been well appreciated by **Software testing professionals** and **UTI** (Unified Testing Initiative, nonprofit organization working for Quality Standards for Mobile Application with members as Nokia, Oracle, Orange, AT & T, LG Samsung, and Motorola). Having started with this column he is also a Core Team Member of **Tea-time with Testers**. Contact Anurag at anurag.khode@hotmail.com

"I am new to Mobile Testing. Please let me know how to get started?"

"I am working since last 6 months in website testing. Due to project need I am supposed to be worked in mobile testing for upcoming days. Please suggest me, how to proceed in this field."

Well, questions like these and many more questions of similar kind, I always encounter on "**Mobile QA Zone**" and "**Mobile Application Testing**". For most of the newbie in this field, these are some of the most important questions, which also define their further progress in this arena.

So, without wasting more time, let us move ahead from this "**Zero Mile**" of **Mobile Apps Testing**.

Some tips you can find below, which may help you progress in the field of Mobile Application Testing.

1. Basics of Software Testing:-

Whether you are experienced or a fresher in this field, Knowledge of Software Testing basics is essential in any kind of testing you perform. Even if you are starting your carrier as Mobile Apps Tester, you need to know all the software testing principles, Software Testing Technique's, Types of Software Testing, Objective of Testing and other basic s in Software Testing. On internet, there are many sites and resources available, which can guide you about the same. If you are just starting your carrier in this field, don't bother to accumulate all the complicated theories of software testing in your mind at once. Go step-by-step .It is strongly recommended to undergo some good training program in software testing which can give you some practical experience of testing .Please avoid mugging complicated testing theories.

2. Basics of Telecom:-

As you have decided to work in Mobile Domain, knowing the Telecom basics will always give you an added advantage. Mobile Apps Testing is not just about testing of Mobile Product or application. You will surely get an advantage if you are aware of other things in this domain which revolves around Mobile products testing. 2G, 3G, CDMA, GPRS, GSM, HSCSD, SIM, SMS, WAP are some basic things of telecom that you should be aware of. You can check out some more details about the same [here](#)

3 Awareness about Mobile OS/Platforms:-

There are many Mobile OS/Platforms present in the market. Android, iOS (iPhone),Blackberry OS(RIM), J2ME, Symbian, Palm, Windows Mobile 7, Samsung Bada ,Nokia Meego and so on. It's very important to have knowledge about these Mobile OS/Platforms as a Mobile Apps Tester. Awareness about the capabilities and limitations of these platforms gives you confidence to differentiate application bug with platform/OS limitations. You may find some more details about Mobile Operating Systems [here](#)

4. Get familiar with your own Mobile Phone:-

I am sure many of you must not even be completely aware of which Mobile Handset model you are using, what is the OS in it and what is the Software Version your phone is having. If you are a beginner, just start exploring your own cell phone. Just open up and try using internet on your Smartphone. Use Wi-Fi, GPRS. Check out how you can format or perform Factory reset on your device. Try to upgrade OS version of your Android device. Experiment with different setting and user permission on your phone. In short, be familiar with all the features and functions your mobile handset is having and it will surely help you dig out more scenarios, while testing any mobile app or any mobile Handset you are given to test it.

5. Get aware of Testing in Mobile Domain:-

When we talk about testing in Mobile domain, it is not only confined to Mobile Apps, but also includes mobile handset and mobile website testing.

Downloadable Mobile Application Testing: - Some applications comes preinstalled in mobile handset ,while some mobile application are downloadable from different mobile application stores(Apple App store ,Android Market, Getjar, Nokia Ovi Store, Balckberry App world etc.) .

Apart from conventional Functional and UI testing, you may need to test your application against the submission criteria's and guidelines provided by these Application stores. As said earlier, tester's role here should not be only functional testing of mobile app, but also to make sure your application is adhering to guidelines provided by these mobile app stores.

Mobile Handset Testing:-

Similar to Organizations that develop third party downloadable mobile applications, there are many companies that develop complete mobile handset. A mobile QA here may need to test native applications or features that are available in the phone .SMS, MMS, Voice Call, MMS, Phonebook, Calculator, Bluetooth and other mobile features. It also includes Multimedia (Camera, Video, Media player, ringtones) and *Mobile Protocol stack testing*.

Mobile Website Testing (WAP Sites):-

Unlike downloadable mobile applications, mobile websites can be accessed via browser. No download involved. Testing of Mobile WAP sites has its own challenges. Proper navigation, good user interfaces (Design), Security, Performance and mobile browser compatibility are important areas.

4 Get Aware of Types of Mobile Apps Testing:-

Similar to general Software Testing, Mobile Software Testing also includes:-

- User Interface Testing (Color scheme, Menu styles, Consistency of UI over various Devices)
- Functional Testing (Testing core functionality of Mobile App as per specification)
- Performance & Stress Testing [Behavior of Mobile Application in Low resources(Memory/Space),Behavior of mobile website when many mobile user simultaneously access mobile website)]
- Usability Testing (Testing of usability aspects of Mobile Apps)

Apart from above mentioned testing types, some key testing types may include the following.

- **Testing for Compatibility:-**Testing Compatibility of your application with native device features (i.e. To make sure your application is not hampering native device functionality)
- **Certification Compliance Testing:-**For downloadable mobile applications, there are various Third party Mobile Quality Certification program for various platforms. **True Brew Testing** (for BREW Apps), **Java Verified program** (for J2ME apps), **Symbian Signed Test Criteria** (for Symbian Apps) are some examples. Apart from regular functional testing, you may need to test your application against the test cases/Testing criteria provided by these certification processes. However, this depends on your client, whether he wants to certify their application these criteria.
- **Submission Guidelines Compliance Testing:** - For application downloadable from mobile app stores, application need to adhere the specified submission guidelines. Failure to meet these guidelines may result in rejection of your app on mobile application stores. For example failure to comply with application **Submission guidelines for Apple App Store** may result in rejection of your app in Apple app store.
- **Interruption Testing** (Voice Calls, SMS, Charger, Low memory Notification) while application is running.
- **Monkey Testing:** - Continual key pad entries via tools to test application stability with various key press events.
- **Low Network/No Network case Testing:** - Application behavior when there is no network coverage or Low network strength.

5 Go through Sample Test Cases for Mobile Application:-

For newbie in Mobile Testing, it is always advisable to go through some sample test cases for Mobile Apps. Going through the test cases for any mobile application, gives a complete insight of the kind testing conducted for Mobile Application. You may ask your seniors to provide you some Test Suites for any mobile project your organization completed earlier. However, you can still go through some general Test Cases for Mobile application [here](#).

6 Explore the Capabilities of Simulator:-

Simulators always play big role when there are no mobile devices available for testing. Though Device testing is always preferred as it represents more likely end user scenarios, the importance of simulators cannot be ignored. In order to have effective testing over Simulator, It is recommended to explore all the capabilities of Simulator.

7 Take Help from Remote Device Access: Service-

Due to large number of devices available in the market, it is not feasible to buy a new device every time. At the same time Simulators are not completely reliable enough to launch a mobile app based on testing conducted on simulators. RDA (Remote device services) can be a good solutions to deal with this challenges. The remote device access services enable access to a live device over the Internet. As a Mobile Apps Tester, you should be aware of such services and should suggest your Manager about the capabilities of such Services.

Some Available RDA Services are:

- **Device Anywhere** (Paid)
- **Perfecto Mobile** (Paid)
- **Nokia RDA** (Free, For Symbian Phones)

Advantages of RDA:

- You don't need to purchase actual device.
- User can select different Carriers for e.g. Verizon, T-Mobile, AT & T.
- RDA'S are more reliable than simulators as they are real devices.
- Some RDA Service like Device Anywhere also has automation capabilities.

Disadvantages of RDA:

- Since you access devices remotely it takes time for any action or key event.
- Sometimes the needed device is not available due to prior reservations.
- Higher Service Cost

8 Explore Tools and Utilities:-

There are many Software Tools and utilities available in the market, which may help you effective testing of your Mobile Application. Some of these tools are found available in SDK'S itself. However, you may still dig out the internet for such tools on various platforms.

- **Tools to check Battery Consumption** while your app is running. For ex. Nokia Energy Profiler.
- **Tools/Software to capture screenshot**: - There are many tools available for various mobile platforms to capture screenshot from device itself. For ex. **Screenshot** tool for Symbian S60 Devices.
- **Tools to Generate dummy files to test behavior of mobile app at Low EFS**. For ex. Maxfilecnt utility from QUALCOMM for BREW mobile apps.

- **Tools to Generate Random key events.** For Ex. [Monkey tool](#) (Android), [BREW Grinder](#)(BREW), [Hopper Test Tool](#) (Windows Mobile)
- **Tools to Capture Logs.** For Ex. [Apphance](#) for Android.

9 Explore Automation Tools for Mobile:-

Along with Manual Mobile Testing, be ready to explore your potential in Mobile Automation Testing. Mobile Testing is a new field and many automation tools are coming in to market gradually. If you get any opportunity to work on any mobile automation tool, it's well and good. But if you are not getting such opportunity, it's better to explore on your own. Don't wait for the time when you will get an opportunity to work on Mobile Automation Tool. Believe me, sooner or later you will definitely face a question from your Interviewer, "Have you worked on any automation tool for mobile?" ☺

Here are some automation tools available for Mobile Apps Testing:-

- TestComplete
- M-Eux
- TestQuest Countdown
- Test Quest Pro
- Robotium
- VNC
- Deviceanywhere
- FoneMonkey (iPhone)
- Eggplant (iPhone)
- TestiPhone(For iPhone Mobile Web)
- IBM® Rational® Performance Tester (RPT)
- 3P Mobile
- Expertest
- MITE (A Mobile content testing and validation tool for Mobile Web app)



10 Explore Communities, Forums, Blogs :-

Even if you work for 24 hours a day in exploring various aspects in this arena, I guarantee, even 24x 7x365 will not be enough. The reason is very simple. You cannot get opportunity to work in each and every area of this domain and by that time you will manage to do that, you will find there are many more things left to learn. Finally Technology is changing everyday and in Mobile Domain, even at faster pace ☺ .Hence, the solution is to join Communities, Forums to learn from mutual knowledge sharing and experiences. Initiate the discussions and you will find there are many QA's sharing their experiences.

Here are some Sites and links in mobile domain, which may help to you.

- ❑ [Mobile Application Testing Blog](#)
- ❑ [Mobile QA Zone-A Mobile/Tablets Testing Community](#)
- ❑ [Nokia Forum and Mobile Test Automation Group](#)
- ❑ [Mobile Apps Testing-Linked In Group](#) [Android Testing Group](#) and [Iphone Mobile Testers Group](#)

In my upcoming articles, we shall take a look on User Interface testing of Mobile Application. Till then Stay Tuned ☺. Do not forget to follow me on twitter-[@anuraagrules](#), [@mobileapptest](#), [@mobileqazone](#) or drop me a mail at anurag.khode@hotmail.com



testing intelligence

- its all about becoming an intelligent tester



an exclusive series by **Joel Montvelisky**

Choosing correctly “What NOT to measure”

***It's not only about what you have in your metrics,
it's also about what you leave outside of them.***

This story begins one day, just like any other day...

The whole R&D department of company XYZ Ltd. is sitting in the Open Space area, listening to an interesting presentation about [Kanban](#) by one of the Team Leads. He is explaining how this process is helping his team to achieve faster deliveries and how it provides better visibility and control over the end-to-end process.

The idea of the Company is that starting from the next sprint (Agile sprint, just in case), all the teams will switch from working under Scrum to some sort of Scramban (Scrum + Kanban) that will allow them to focus on a limited number of tasks, in order to develop their product faster and with more visibility to all the company stakeholders.

The last slide of the presentation is about the 5 things that are “very important” to **measure**, to ensure the team is working under the principle of *Kaizen* or continuous improvement:

1. WIP (Work in Progress)
2. Lead Time
3. Waste (efficiency)
4. Throughput
5. Quality

And here is where the session turned interesting. When someone in the crowd (someone, for example Me) asked the Team Lead how do they measure Quality in his team? And then all hell broke loose...

Tell me who I am and I will tell you where to find me

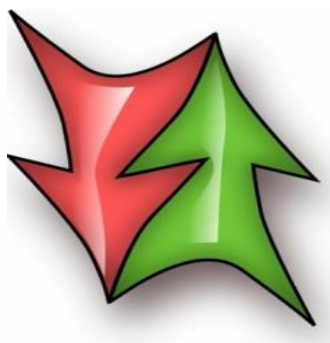
The question may sound simple at first – *how do you measure quality?* – but if you ask 10 persons from different teams in your organization to define Quality, you will find between 7 and 15 different definitions (some people will not be able to provide you only “one” answer...). So, who’s Quality will you choose to measure?

Let’s move further down this road, when I asked the forum of Development Team Leaders, once the session was over and we got together to review this point in more detail, we were able to reach the conclusion that we wanted to measure Quality by taking a look at the bugs (bugs are not the only indication to measure quality, but it is usually the simplest and fastest way to start measuring!).

But then we immediately started arguing about what bugs to measure? Bugs reported “as part of the development”, bugs detected “once the product was released”, or bugs “detected AND corrected once the product was released”...

In short, everybody has an opinion and all of them are valid in one way or another.

What you measure may improve, what you don’t measure will always fall behind



The biggest problem with metrics is the psychological effect it has on the people being evaluated.

Start measuring something and you will see how it improves:

- How far or fast you run.
- How much food you eat.
- How many hours a week you spend with your wife and family.
- How many bugs you find or tests you run. etc

But since your resources and your capacity are limited, then for everything you improve there will need to be something else that “pays the price” for it:

- You will be able run farther or faster because you train a lot, but your grades at school may come down because you don’t have enough time to study for your tests.
- You will eat less, but you will also smoke more because you are always hungry.
- You will spend more hours a week with your wife and family (and you will still work the same amount of hours because you don’t want to get fired!), but you will have less friends since you don’t have time to spend with them anymore.
- You will find more bugs, but more of your bugs will be rejected as “not clear enough”, as “unable to reproduce” or as “duplicates” because you spend less time writing them and reviewing them.

- You will run more tests, but since you don't have more time to run them they will be less complex and won't represent the user's scenarios good enough, so in the end you end up releasing more bugs to the field.

What's the solution? Is it to stop measuring? Definitely NO!

The solution is not to stop measuring, as I said before measurements are one of the most effective ways or tools you have to improve things. But you need to be SMART and take into account that it's not only about what you have in your metrics, it's also about what you leave outside of them.

For example, if you start measuring the speed with which you develop each of your features make sure you also measure the quality of your deliverables. Take also into account that if you develop each feature faster and don't harm the level of quality, then you will most probably need to develop less features overall – at least at the beginning of the process.

The same goes for every other measure you can think of! There's no free lunch, someone or something will always need to eventually pay the bill (or wash the dishes...)



Define an acceptable price you are willing to pay up-front :

A good way to go around when you define a "Set of Measurements" for your team is to take into account the following 3 principles:

1. Keep your metrics **simple**, easy to **calculate**, and logical to **understand**.
2. **Don't use a single metric**. Define a set of measurements with the intention of covering your process and your needs from 4 or 5 different and complementary angles. On the other hand don't set more than 4 or 5 metrics at a single time!
3. Provide **guidance** to your team, either by stating this in writing or by making this common knowledge, about **the price you are willing to pay** in order to improve your measurements. In simple English: What other aspects of your operation will you be willing to harm in order to achieve improvements on the measured areas?

By doing this you are making sure that all your team is in sync and this will increase your chances of really improving your process, and maybe more importantly of getting the correct feedback needed in

order to modify your process (and your measurements) quickly in case they are harmful or counter-productive.

Got any experiences to share? Feel free!

Maybe you have some interesting stories to share about metrics, please feel free to add them as comments!

From my experience, many of us have "lived through" metrics-abuse-catastrophes in one or more occasions, so I'll be happy if for a change some of you have been part of experiences or have insights about situations when metrics were used smartly (maybe also in an unconventional way!) and were able to really make a difference...

Biography



Joel Montvelisky is a tester and test manager with over 14 years of experience in the field.

He's worked in companies ranging from small Internet Start-Ups and all the way to large multinational corporations, including Mercury Interactive (currently HP Software) where he managed the QA for TestDirector/Quality Center, QTP, WinRunner, and additional products in the Testing Area.

Today Joel is the Solution and Methodology Architect at PractiTest, a new Lightweight Enterprise Test Management Platform.

He also imparts short training and consulting sessions, and is one of the chief editors of ThinkTesting - a Hebrew Testing Magazine.


Joel publishes a blog under - <http://qablog.practitest.com> and regularly tweets as [joelmonte](#)

Back To Index



by





Call for Articles !

Have you got something to say?

yes, we are listening you...!!!

"Tea-time with Testers" firmly believes that one of the best ways to improve upon software testing is to listen to the lessons learned by others and their experiences too.

So, if you have an interesting story that you'd like to share with the world, contact us at teatimewithtesters@gmail.com.

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

T ' Talks



T. Ashok exclusively on software testing

Musings & Inspirations: Learning from Non-Software Disciplines

As a moderator of the panel discussion on "Roadmap to higher quality" at **SoftTec Asia 2011** at Bangalore, with three panelists from non-software disciplines - Pharma, Food/Beverages and Automotive industry, my job was interesting, of seeing parallelisms and looking for inspirations from other disciplines.

The panelists were from the quality assurance department and they talked about how they assure quality in their disciplines. They talked about standards/regulations, design stage behaviour analysis, statistical sampling and interesting ways to uncover defects.

As a keen observer of their ways of doing and reflecting on our practices, I discovered some interesting parallels and as inspired into look into software testing in different ways. I have captured my musings and inspirations that you should find interesting.

Goal-focused versus activity-based standards

These industries are regulated and the product has to comply with standards before they are released. Hmmm... we in software also have standards too. But there seems to be some key differences. The standards that we follow are "prescriptive"; they describe the steps that one has to do. On the other hand, the standards in the above described industries are probably "descriptive".

They state as to what is expected in the product, they set a baseline to comply with. They communicate clearly as to what is expected, in what is termed as 'Cleanliness criteria's in HBT (Hypothesis Based Testing).

The big difference is that whilst we; in software are focused on activities and ensuring consistency of these, our friends are focused on the outcome or the goal. BIG DIFFERENCE.

Staining the system to see defects

To uncover defects in software, we design test cases which when executed uncover defects. In the beverage industry, there is an interesting way of uncovering defects. Here used bottles are cleaned and filled with the beverage. So how do we know the bottles are indeed clean? My colleague from the Food & Beverage industry mentioned an approach that I found very interesting.

When bottles are cleaned, the expectation is that moulds shall not be present. But, how do we check for presence of moulds, as they are not visible to the naked eye? They use a test called Dimethylene Blue (DIB) Test, where DIB solution is poured into the bottle. Voila, the moulds, if present show up as blue stains, which a trained eye can detect. Cute eh!

Would it not be interesting to if we had a staining solution to 'stain the defects in software'? What an idea Sirjee!

Let's us continue the train of thought and see if we have any parallels. When we insert assertions into code, it flags faulty code when violations occur. This is kind of like the staining liquid, but this is at a low level or deep inside, and requires a good programmer to lace the code intelligently with the 'staining liquid'.

As of now, we do not seem to have an approach like at a higher level that can be done with less intelligence. How about memory leak detection? Hmm.. this seems to employ this idea. Here we are looking for a specific kind of a defect (a mould), where the leak detection tool detects stains (memory leaks) without requiring any specific intelligence from developers.

Would it not be interesting to invent a universal staining solution that when applied to software can make defect visible?

Early stage design behaviour analysis

In non-software disciplines, an enormous effort is expended in design before the product is constructed out of real materials. It is indeed necessary to do so, as the cost of rework at manufacturing phase is very expensive. Think of this as "low malleability", the cost of morphing (I.e changing) at the late stage of manufacturing is very expensive. We probably believe that software is more malleable. Is it?

In other disciplines, the design is checked dynamically, by simulating behaviour whereas most often, we assess design statically via design reviews. The designs in automotive industry are subject to serious dynamic analysis of behaviour. Finite Element Analysis, Crash Analysis are some examples where the dynamic behaviour is examined formally, before the product is manufactured.

Statistical sampling & analysis to build confidence

Statistical sampling is rigorously applied to check the manufactured items and build confidence in quality. The question however is- are there activities that we do, that could be similar to manufacturing? Software development is understood as a creative activity and therefore never compared to manufacturing.

Let us think on us... When multiple items are being manufactured, the premise is that variations can occur and these can result in certain items being defective. Whilst we do not seem to have a manufacturing approach, do understand that our key resource is the intellect, with the same person

developing various pieces of code for a project. So is it not probable that a person could be possibly creating similar types of defects?

If so, could sampling of types of defects in various code developed by a person prove to be useful? Could this give an insight into our defect injection capability and therefore improve defect detection? Could this also act as our personal mirror allowing us to prevent defects?

Serious focus on attributes

Non-software industries seem to have evolved to a higher degree compared to our industry, with strong focus on non-functional aspects or attributes. They have a strong focus on reliability, stress handling, safety, endurance, stress and usage.

We in software do focus on attributes also, however I felt that our colleagues from non software industries expend enormous effort on these and commence evaluation of these real early. On the other hand I think that we expend a considerable effort in evaluation of functional correctness. Could thus be our colleagues are helped by standards that state attributes clearly? Hmmm...

Closing note...

The quest for perfection and pursuit of excellence has been consuming mankind from time immemorial. That said, every discipline looks at the act of delivering excellence slightly differently, and examining these with open mind provides us with inspirations and innovations to a cleaner software world

Cheers! Have a great day.

I would love to hear your comments/views. Tweet me @ash_thiru.

[Back To Index](#)



Biography



T Ashok is the Founder & CEO of STAG Software Private Limited.

Passionate about excellence, his mission is to invent technologies to deliver "clean software".

He can be reached at ash@stagsoftware.com.

A black and white photograph of a man in a suit and glasses looking through binoculars. The man is looking upwards and to the right. The background is a cloudy sky. The text 'Tool Watch' is overlaid in a large, white, sans-serif font.

Tool Watch

about various testing tool around

StressTester™

PART 1

Tea-time with Testers Rating: ★★★★★

by **Juhi Verma &
Sharmistha Priyadarshini**

Introduction :

StressTester - is an enterprise class performance testing tool from **Reflective Solutions** to reduce project timescales and costs while ensuring 'Correct & Realistic Testing'.

It has features and functionalities needed to perform load, stress, spike, and soak application performance tests without needing scripting skills. StressTester is used by global organisations to verify the performance of their mission critical systems.

Supported Platforms

StressTester is certified to run on the following platforms:

- Windows – XP, 2005, 2008 and Windows 7
- Solaris – version 8 and later
- Linux – all common distributions
- MAC/OS – version 9 and later

** StressTester™ is written in Java and should run on any operating system that supports Java. Please contact Support if you would like help with running StressTester™ on a non-certified operating system.*

Pre-Requisites

StressTester requires:

- Java version 6 or above
- 20Mb of disk space



Underlying Database

StressTester uses a relational database to store configuration information and performance test results. The software is supplied with an embedded database – Apache Derby – but this is only suitable when trialing StressTester.

If you wish to use StressTester for performance testing you will need to choose either SQL Server or Oracle as your underlying database.

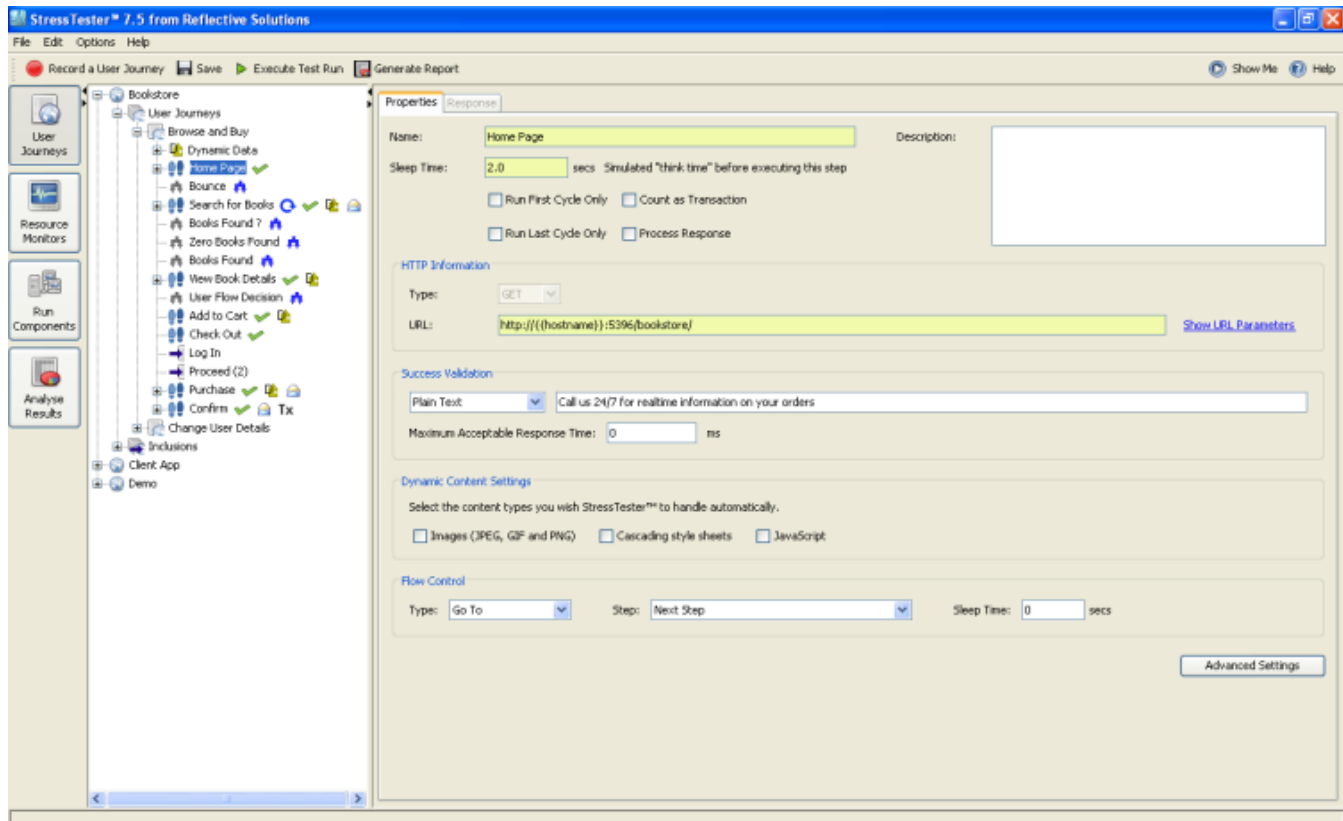
GUI Workspaces

There are four workspaces within the GUI; each is concerned with a different aspect of the performance testing process.

Common to each workspace is a navigation panel and a display/editing panel. By selecting a node in the navigation panel, its data is displayed in the editing panel and can be viewed and/or updated.

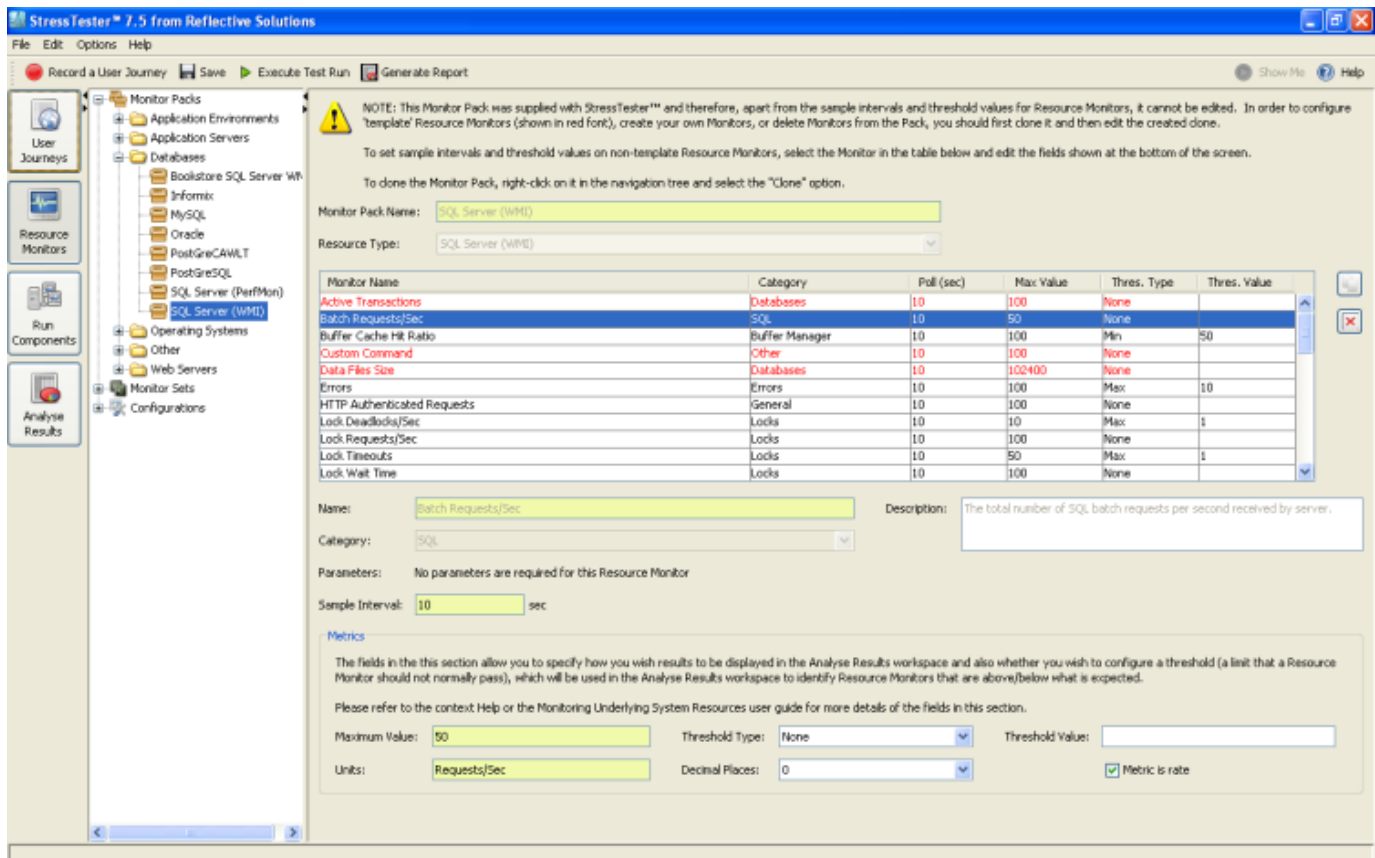
1) User Journey Workspace:

The User Journey workspace is used to record and configure User Journeys and Inclusions.



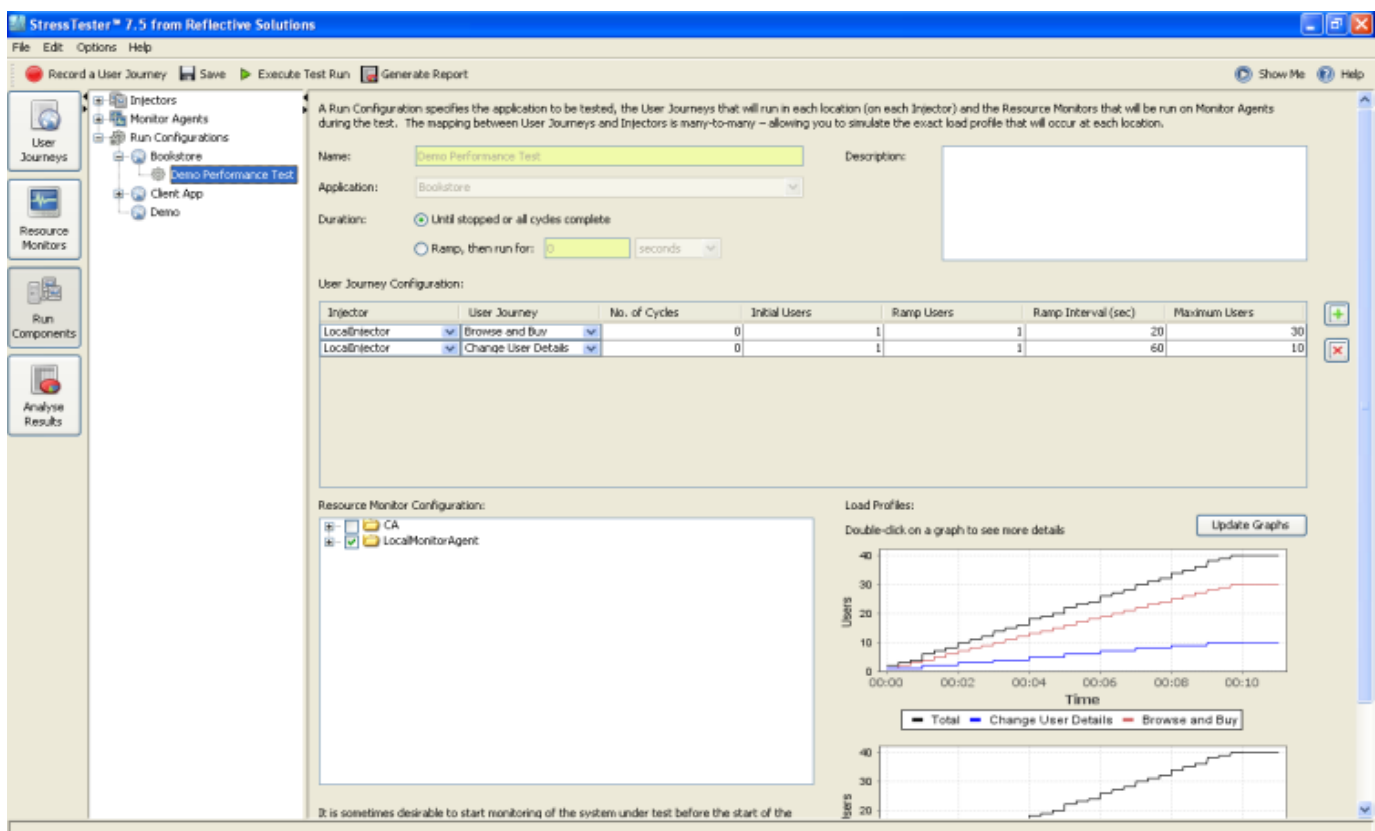
2) Resource Monitors Workspace:

The Resource Monitors workspace is used to configure Resource Monitors and create Monitor Sets and Monitor Configurations.



3) Run Components Workspace:

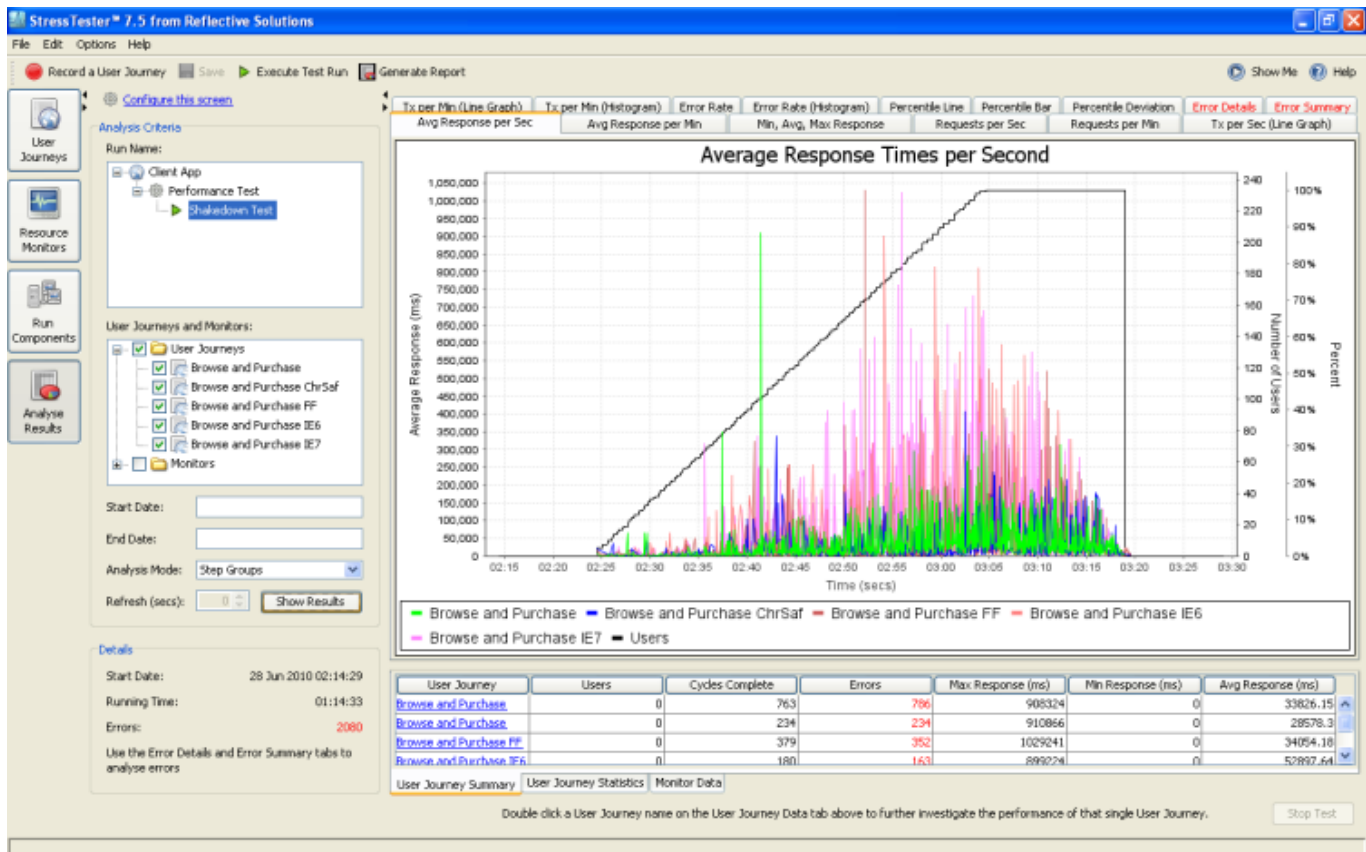
The Run Components workspace is used to configure Injectors and Monitor Agents and create and edit Test Run Configurations



4) Analyze Results Workspace:

The Analyze Results workspace is used to analyze the results of currently executing performance tests (in real-time) and previously executed tests.

Application response times can be correlated with Resource Monitor results to determine the cause for poor performance or other identified issues.



GUI Conventions

- Mandatory Fields**

The majority of edit screens within the GUI have one or more mandatory fields.

These are identified by their yellow background. In the example below, the Name, Sleep Time and URL parameters are mandatory.

- Right-Click Menus**

To perform an action on a node selected in a Navigation Panel, right-click on the node to display a node-specific list of options.

All options on the toolbar Edit menu are also available using right-click on the relevant node.

Planning a Recording

StressTester allows you to vary the data supplied to the application and the route(s) taken through the simulated business transaction by configuring the User Journey after it has been recorded.

Simply ensure that you visit each page that may be in a User Journey route during the recording and don't worry if you forget pages; you can add them to your User Journey at a later time.

Starting a Recording Session

To start a recording session, click the  button in the StressTester GUI.

User Journey Details

The recording wizard will start, and after the Introduction screen, you will be asked to enter a Name and Application for your User Journey and optionally a description (User Journeys are grouped by application within the User Journey workspace).

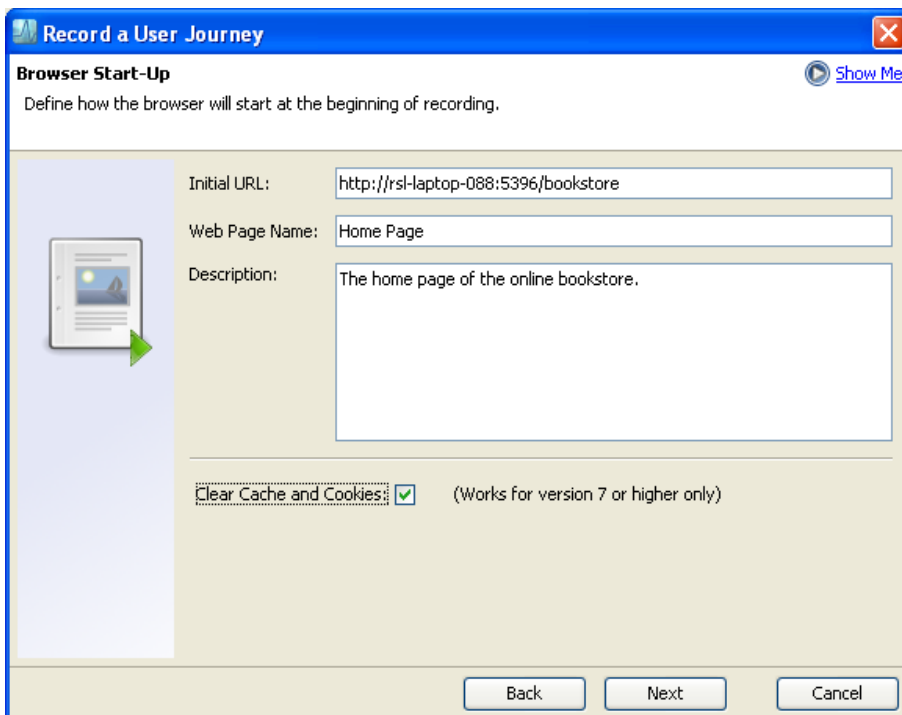
Browser Selection

When you click 'Next' you will be prompted for the browser type you would like to use during the recording.

StressTester integrates with Internet Explorer and Firefox.

Initial Start-Up

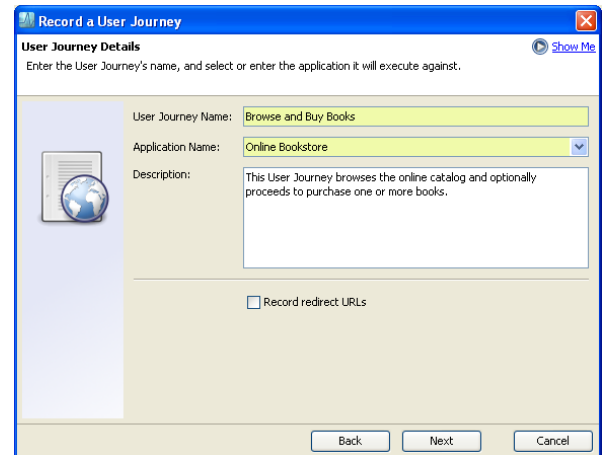
Next you have the option to enter the URL you would like to use to start your recording, a name for the corresponding page (its name within the User Journey), and a description.



The 'Record a User Journey' dialog box, 'Browser Start-Up' tab, is shown. It contains the following fields and options:

- Initial URL:**
- Web Page Name:**
- Description:**
- Clear Cache and Cookies:** ☒ (Works for version 7 or higher only)

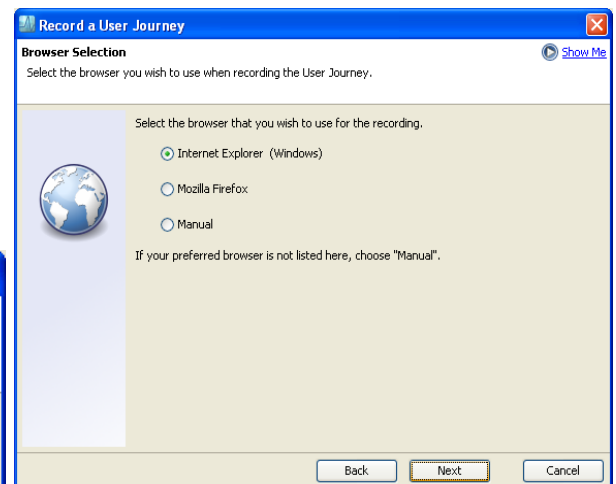
Buttons at the bottom: Back, Next, Cancel.



The 'Record a User Journey' dialog box, 'User Journey Details' tab, is shown. It contains the following fields and options:

- User Journey Name:**
- Application Name:**
- Description:**
- Record redirect URLs:** ☐

Buttons at the bottom: Back, Next, Cancel.



The 'Record a User Journey' dialog box, 'Browser Selection' tab, is shown. It contains the following options:

- Select the browser that you wish to use for the recording.**
- ☒ Internet Explorer (Windows)
- ☐ Mozilla Firefox
- ☐ Manual

Text below: If your preferred browser is not listed here, choose "Manual".

Buttons at the bottom: Back, Next, Cancel.

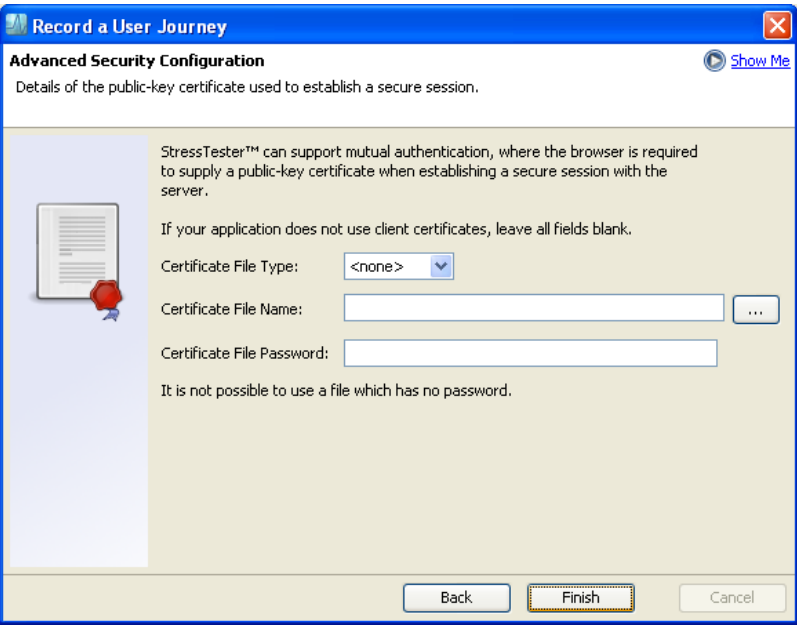
As you visit web applications, your browser will cache content so that it does not need to request this again. In addition, the applications may send cookies to the browser that change the application's behavior (for example not prompting the user to log in).

If you wish to simulate users that have previously accessed the application recently, then you can leave the Clear Cache and Cookies option unchecked. Make sure you also run through the User Journey in the browser at least once before recording, in order to warm up the cache.

If you wish to simulate users that have not recently accessed the application (or are visiting for the first time), you should check the option.

Advanced Security Configuration

When you click 'Next' you will be presented with the option to specify a digital certificate file.



This is only necessary if you are testing an application that requires a digital certificate to authenticate the user.

Clicking 'Finish' will close the wizard, open the selected browser and, if you supplied one, the browser should display the initial web page.

Annotation Window

During the recording session, the Annotation Window will be open and 'always on top'.






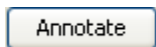
The Annotation Window enables you to start to configure the User Journey during the recording; providing names for actions and web pages, entering notes, and determining -

what is counted as a transaction (for when StressTester calculates transaction throughput information).

Do not worry if you forget to do something in the Annotation Window during your recording; everything you can do in the window is also possible within the StressTester GUI at any time following the recording session.

The buttons in the Annotation Window are described below:

	Recording status. When flashing this indicates that the requests from the browser to the application are being recorded.
	Pause. Clicking this will pause the recording but you can still use the browser. This is typically used when you are re-recording parts of existing User Journeys (to replace parts of an existing User Journey).

	Restart. Click to restart recording when you have previously paused it.
	Stop. This will end the recording session and start the Post Recording Wizard .
	<p>Mark as Step Group. When selected (default option) every newly annotated request will be made the lead step within a Step Group in the User Journey.</p> <p>When testing web applications, this means the annotated step will be the main URL of a web page.</p> <p>If you wish to annotate steps without creating a new Step Group, unselect this button, annotate the step, perform the action in the browser and then select the button again before annotating the next Step Group.</p>
	Mark as Transaction. If selected, the annotated step will have Mark as Transaction set in its properties and will be counted as a transaction when StressTester™ calculates transaction throughput statistics.
	Opens and closes the Description field allowing descriptive text to be entered for the next step to be annotated.
	Click to annotate the next request with the name entered and any other details set using the other buttons and fields.

Recording Process

There is an easy process to adopt to ensure that every time you record a User Journey, your recording contains all the information you need.

The first stage is to plan the recording.

Then, for every step within the plan, you should “annotate then action”.

So, if recording a step to search a books website for testing books, you would:

1. Enter the Step Group Name “Search for Books”
2. Add a description of “Searched using term ‘testing’ ”
3. Check that the Step Group button is selected
4. Click Annotate
5. Perform the search in the browser
6. Wait for the browser to show that the request has completed

Repeating as above for every step in your plan will ensure the User Journey recorded matches the plan.

To be continued in Next Issue



Biography



Juhi Verma is an Electronics Engineer working with Tata Consultancy Services (Mumbai) as a Tester. During her spell she has also worked as a programmer but she now prefers software Testing over programming as it's a dual fun, she says.

Testing is her passion and she enjoys participating and conducting testing related activities.

Juhi also works as a Team member at Tea-time with Testers. She can be contacted at her personal mail id juhi_verma1@yahoo.co.in or on Twitter @Juhi_Verma .



Biography



Sharmistha Priyadarshini is currently working as a Test Engineer at Tata Consultancy Services (Mumbai).

Sharmistha is die hard lover of Data Base testing as she finds it challenging. Being a programmer in past she now loves software testing too as it gives her more scope for analysis. Sharmistha can be reached via her mail id sharmistha.priyadarshini@tcs.com

Do you think that even your own tool should be part of this unique section?*

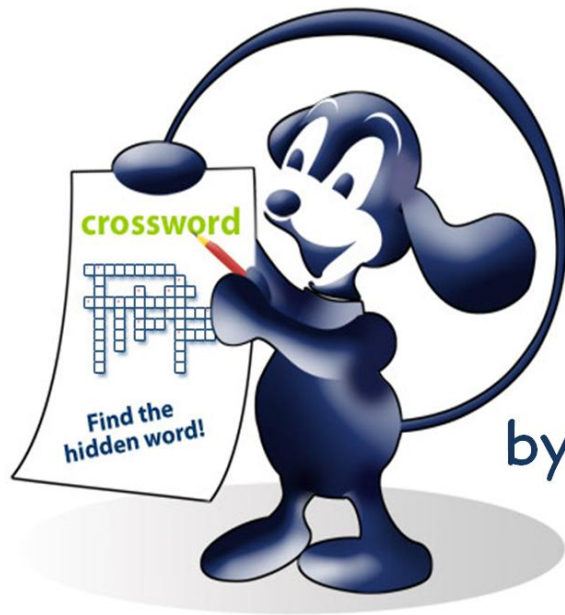
Feel free to write us. Let the world know what your Tool can do !

To know more write to us at teatimewithtesters@gmail.com

* Conditions Apply

Testing PUZZLES

by Sebi



Claim your **Smart Tester of The Month Award**. Send us an answer for the Puzzle and Crossword below by 16th Aug 2011 & grab your Title.

Send -> teatimewithtesters@gmail.com with Subject: Testing Puzzle

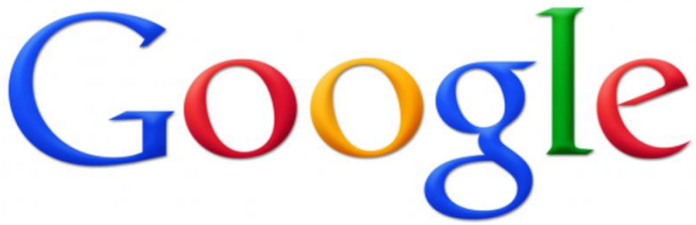
Gear up guys.....

It's Time To Tease your Testing Bone

Puzzle “Google it Out”

"What is the shortest string that you can use in Google search to get 0 results?"

Eg. This string "sdfsfadfasdfzzzz333333333242" doesn't return any search result . 😊

The Google logo is displayed in its standard multi-colored font.

Biography



Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years. He is currently hosting European Weekend Testing.

He considers himself a context-driven follower and he is a fan of exploratory testing.

He tweets as @testalways.

You can find some interactive testing puzzles on his website www.testalways.com



TESTING CROSSWORD



1	2			3			4			5	
6					7		8				
9			10		11						12
					13				14		
	15										
									16	17	
	18				19	20					
					21						
	22						23				

Horizontal:

1. Software is modified for any reason testing needs to be done to ensure that it works as specified and that it has not negatively impacted any functionality that it offered previously (11)
6. Method used for creation of _____ object in QTP (6)
8. He is the author (first name) of QuickTest Professional Unplugged (5)
9. This is one of the recovery scenario (second word) in QTP and useful in handling crashed applications at runtime (5)
11. AND, OR & NOT are known as _____ operators (7)
14. Accessibility Check point is used to check the _____ standards on web pages in web application (3)
15. Accunetix is the most popular _____ testing tool (8)
16. It is a scripting language to use in Winrunner (3)
18. Testing done on code (8)
21. How To Run QTP Scripts at Scheduled Time? The third word of this sentence (3)
22. The wait statement in Silk Test Tool (5)
23. It is an online test case management (TCM) and test-tracking system built with PHP and a SQL backend tool (short form) (3)

Vertical:

2. Deviation from software requirements to design(5)
3. It is a tool (short name) to allow Java programmers to write re-usable tests for web applications that, unlike HttpUnit, drive the actual web browser on the actual platform they intended to support (3)
4. Logical extension of unit testing (11)
5. A piece of code that simulates the activity of missing components (4)
7. It is a quick and easy command line automation tool, in short form (3)
9. It is a methodology (short form) used to develop and refine an organization's software development process (3)
10. Every project/product involves this life cycle (4)
12. Opposite of Global Variable in QTP (5)
13. Defect Found by Internal Team/Total no of defects found) * 100 is called, in short form (3)
14. It is a open source web automation testing tool which uses Watir as the library to drive web pages (3)
17. An automation tool (first name) for testing the functionality of enterprise applications in any environment, it is invented by Segue Software and acquired by Borland in 2006 (4)
18. Service built for a Web Application is called, in short name (3)
19. Oracle, SAP _____ tools (3)
20. An error in a program or a malfunction in a program's code (3)



Every Tester

who reads Tea-time with Testers,

**Recommends it to friends and
colleagues .**

What About You ?

Our Testimonials

Hi Lalitkumar,

This looks like a very good e-magazine - congratulations! I have only just scanned part of the June issue, as I haven't heard about it until Fiona told me at the BCS SIGIST this week. Having Jerry Weinberg as a contributor is fantastic too!

I would be honored to contribute something at some time.

Keep up the good work. I will also have a link put to the magazine from my web site.

Regards,
Dorothy Graham

Dear Dorothy,

Thanks a bunch for appreciating our work and considering us worth of putting there on your website.

-Editor

I am in testing field from last 10 years but never saw the passion which you guys have for testing.

I went through your website and Tea time with Testers Magazine, and I am surprised to see the way the people are now interested in Testing and the information provided by you is well appreciated

I too would like to join your team and share my experience of testing.

-Sunil Babar, India

Hi, I heard a lot about you, so subscribing it here.
Thanks-

- A

Thanks for developing this kind of ezines for software testers. And Thanks a ton for free subscription.

-Sunil Parihar

I think Tea Time with Testers will be a good platform for the testers to learn new things. Keep up the good work.

- Chetan

Lookin forward to learning and reading about other testers and sharing info

-Hilton

It's a gr8 interface to know new & different concepts of mobile testing

- Sonal

Hi,
This is a great effort from your side. I would also like to contribute to this, if possible, in whatever way I can .

-Arvind Verma

"This is Naresh. A full-time professional tester. Looking to accumulate and learn a lot from here."

- Naresh

Hi,
Really it is a wonderful e-magazine & a very innovative. I got to learn many new things.

Hopefully, I will also be able to contribute to this e-magazine in the future. All the Best !

-Sangram Desai

Looking forward with lot of expectation as I am a Tester by profession and leading a team too so this will be a great eye opener and refreshing apart from being engrossed in our own projects :)


- Purnima Bose

I really enjoyed reading this e-magazine. Great work guys !!

- Anushree Mehra

You ask...

We'll help...!



If you have any questions related to the field of Software Testing, do let us know. We shall try our best to come up with the resolutions.

- Editor

Feel free to write us your expectations.
Help us to help you better.

We are just a mail away: teatimewithtesters@gmail.com

in ne>xt issue

articles by -

Jerry Weinberg

T Ashok

Joel Montvelisky

Anurag Khode

and others

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover Page - CountryLiving.com

Core Team:

Kavitha Deepak (Bristol, United Kingdom)

Debjani Roy (Didcot, United Kingdom)

Anurag Khode (Nagpur, India)



Anurag



Kavitha



Debjani

Mascot Design & Online Collaboration:

Juhi Verma (Mumbai, India)

Romil Gupta (Pune, India)



Juhi



Romil

Tech -Team:

Subhodip Biswas (Mumbai, India)

Chris Philip (Mumbai, India)

Gautam Das (Mumbai, India)



Subhodip



Chris



Gautam

*// Karmanye vadhikaraste ma phaleshu kadachina |
Karmaphalehtur bhurma te sangostvakarmani //*

To get **FREE** copy ,
Subscribe to our group at

Google™

Join our community on

facebook.

Follow us on



www.teatimewithtesters.com

Give Feedback

