THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

JULY 2012 | YEAR 2 ISSUE VI

If you are reading THIS

then you are one of those

17,900 smart testers

who regularly read

TEA-TIME WITH TESTERS

Subscribe here Right Away to get our all Issues for FREE

TEA-TIME WITH TESTERS

First Indian Testing Magazine to reach 97 Countries in the world !

Created and Published by:

Tea-time with Testers.
Hiranandani, Powai,
Mumbai -400076
Maharashtra, India.

Editorial and Advertising Enquiries:

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

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

Editorial

Let's change things for better!

Dear Readers,

A day before yesterday, I got a call from friend of my friend. He said that he has been reading 'Tea-time with Testers' for quite some time and he found it very interesting. This guy works in some organization where he is the only tester. Good thing is that he is his own boss. That means he has complete freedom to perform his testing the way he wants as long as he is giving good results.

Now, you must be wondering why I am sharing this with you. Well, it's because I found his attitude worth appreciating. He told me that there is no one to review quality of his deliverables and he feels restless for not having anyone to give him feedback. He asked me if I can give him feedback for some sample test reports he created or to suggest some people who will help him with the review. He explained me about the efforts he is taking to improve his testing skills. Not only this, but he also told how he convinced his manager to get rid of (time consuming) test reporting methods.

Friends, this *attitude* is something that every tester must possess if at all he/she wants to be great at testing. I also appreciate his manager who gave him chance to put forth his ideas.

Now, some fellows might say that, 'We don't have freedom like him. We don't think our managers will listen to us'. I understand that it's little tough but if you really want to see things changed then it's *you* who will have to work for it. You need to become *restless* for enhancing your skills. Learn things on your own, help your colleagues by sharing your knowledge and don't be shy to propose your ideas. If your ideas make sense, your manager will surely appreciate them. What all you need is just a *desire* to do it.

A '*desire to change*' is where it will start from.

Good news is that there are people who are actually changing things for better. It might interest you to know how 'Barclays Capital's GTC' has brought leading change in community of testers.

It will be interesting to see things taking good shape in future. I am eager to see that happening soon. What about you?

Sincerely Yours,

- Lalitkumar Bhamare

editor@teatimewithtesters.com

June 2012

What a fine issue !

Dear Lalit,

Got your June issue. And what a fine issue it is. Lots of practical advice, along with theoretical underpinnings.

'Understanding Browser Compatibility Strategies' is a terrific article, and might be even better if it said something about how testers could make certain efforts to see that the number of different browsers in use was reduced, thus simplifying testing.

'Use "Cow Magnets" to solve your biggest challenges!' is a great metaphor, and might benefit from some tips on how to keep "metal" out of the hay in the first place.

Similarly, 'Do less work - Test case immunity can help' might benefit from some clues as to how to prevent immunity in the first place. (I first wrote about such immunity about 50 years ago, in Weinberg, Gerald M. "Natural Selection as Applied to Computers and Programs." General Systems 15 (1970 (1967)).

BTW, I really liked the glossary of acronyms at the end of Software Performance Engineering.

Again, thanks for another great issue.

- **Jerry Weinberg**

'Teach-Testing' campaign is brilliant initiative.

I absolutely agree! Especially during the time I spent pursuing network engineering, if such courses were offered then, I certainly would have taken advantage of it.--it has taken me awhile to "find my niche" and something that I truly enjoy.

Certain ideas from my end-

(1) Begin with "Manual Testing" and teach the foundation and include the standards.

(2) Incorporate "automation"; it is essential to augment their knowledge & employability.

(3) Include communication skills--both written & oral; if you cannot write and/or verbally describe an incident/bug using proper language, you will never be taken serious.

(4) Include Team Building sessions involving completing assignments together in groups. This will provide practice for the real world, when working with developers, BAs & other QAs will be really important.

(5) Include Time Management Techniques.

(6) Simulate "Mock" Reviews of Requirements.

(7) Turn Specifications into Test Cases.

(8) Keep Testing ethical.

I find that testing is an art and one must certainly be passionate about it to do it well. I wish you all the best in your efforts.

Kindest Regards,

Gail

TO SEND YOUR LETTERS:

WRITE TO US AT –

editor@teatimewithtesters.com

QuickLook

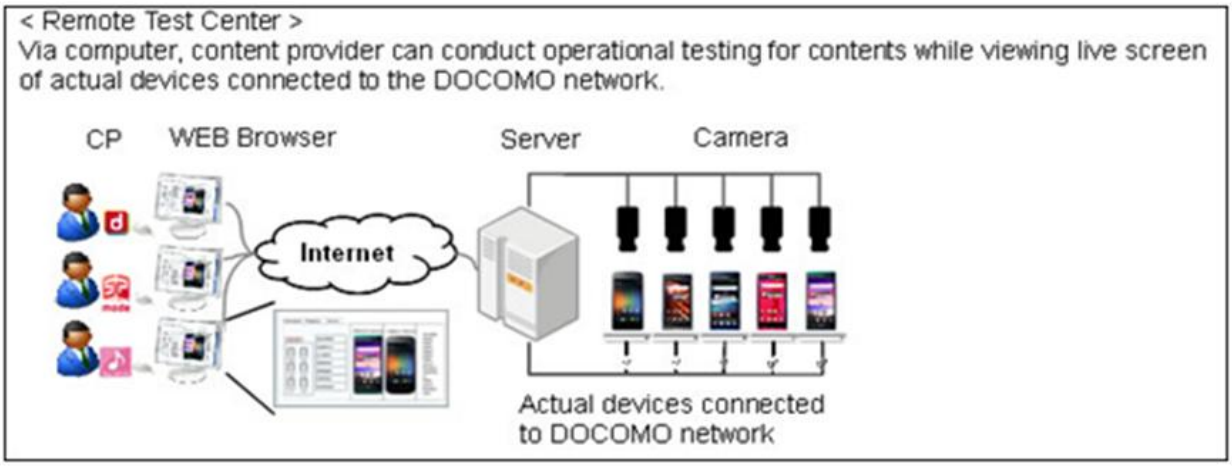
Docomo to Launch Remote Testing Center to Fight with Android Device Fragmentation

July 25, 2012 — anuragkhode

In Recently held conference on 15–02-12,NTT DOCOMO has decided to introduce an elaborate remote testing center, located at the University of Aizu in Fukushima Prefecture, Japan to help developers test their content. This Remote Testing Solution which is based on Perfecto Mobile Platform and executed by Accenture will have hundreds of handsets of all software versions, screen sizes and will help developers combat with Android device fragmentation. Here are some highlights of this proposed remote testing center:-

- App developer can upload their application on the remote devices and can test their application
- The system will allow 60 handsets to be tested at one time
- Developers will be able reserve time slots on specific handsets
- Developer can upload and test their software on this platform
- Developers will be able to run automated batch tests.
- Developers will have Android testing Interface through which they can take desired action such as swipes, taps at specific locations and button presses.

- Remote testers will be able to use the Android testing interface, which allows for actions
- More advanced inputs, like pinching on the touch display, or GPS and accelerometer readings, will not be accessible
- This service will enable smooth portability of content adaptation across devices.



Mobile testing center powered by the MobileCloud platform

In short this solution will be similar as Perfecto Mobile cloud or Device Anywhere from where user can remotely test their application.

Courtesy- mobileappstesting.com

---

qTrace Bug Capture Tool to Integrate with Fog Creek's FogBugz

qTrace releases version 2.5 of screen capture tool

QASymphony

QASymphony (qasymphony.com), the developer of qTrace, an intelligent defect documentation Quality Assurance tool that helps testers create detailed and informative defect records, today announced a reseller agreement and tools integration with FogBugz, a popular defect tracking system developed by Fog Creek Software. Fog Creek will market the qTrace tool to its community of users at a special discount.

More powerful than using simple screenshot or video for documenting defects, qTrace records all steps, screens, and system information associated with a defect. Early users found it cuts defect reporting time by 30-50% and decreases fix time by 10-30%.

New York City-based Fog Creek (www.fogcreek.com) offers a bug tracking system called FogBugz, which tracks bugs, issues, and customer support tickets through every stage of the development process and is used successfully by more than 20,000 software developer teams.

In a matter of minutes, a user can be up and running with qTrace and seamlessly integrated with FogBugz.

The FogBugz platform makes for an ideal integration with qTrace because of its ability to not only track bugs, but to manage projects with issue tracking, scheduling, advanced project management and customer support features such as email routing and discussion groups.

"Upon first using qTrace it became immediately apparent that it would fill a need that no other defect capture product we had found has filled, and it does so in an easy-to-use, intuitive way," said Sam Vanderpol, Director of QA at Fog Creek Software. "We were excited when qTrace decided to build a tight integration with FogBugz, so our customers could get the same benefits we were seeing."

QASymphony also announced version 2.5 of its popular qTrace tool for testers and software developers. qTrace expedites the QA process, helping software development teams bring products to market by as much as 20% faster and reducing defect reporting time by about 70%.

qTrace 2.5 complements its existing capabilities by adding four different single screen capture modes, eliminating the need for the "PrintScreen button" and other screen capture software.

Newly released to the market in Feb of 2012, qTrace has created a new category of screen capture. qTrace not only captures images of software screens, but also the user's actions such as mouse clicks and entered text. By leveraging the screen images and generated text narration, a tester can describe and submit software defects in full detail quickly and easily.

Since its release, qTrace has been well received by the testing community and has quickly responded to feedback by adding enhancements to the tool.

"Our goal is to make qTrace the only screen capture tool needed by testers," says Vu Lam, CEO of QASymphony. "We've tackled the tough part, which is creating a tool with the intelligence to document long complex defects. But we've heard our users say that for simple cases, one screenshot is enough. We've now tuned qTrace to have robust functions in the capture of a single screenshot, so qTrace is the only tool needed to document simple and complex defects. We're also adding some other goodies that I'm sure our fans will appreciate!"

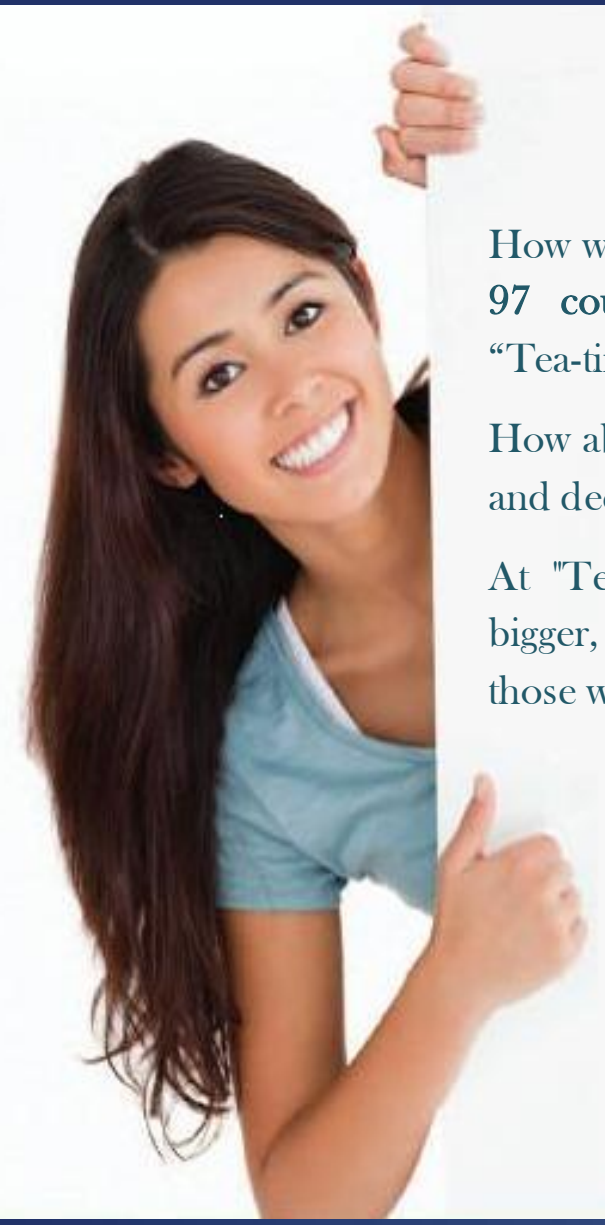
The Record toolbar is redesigned to let users easily switch between different capture modes: recording multiple screens and actions (record session) or just capturing a single screenshot.

Website: www.qasymphony.com **Facebook**: www.facebook.com/qasymphony **Twitter**: www.twitter.com/qasymphony

Press Contact: Victor Cruz , Principal, MediaPR

~ For more updates on Software Testing, visit Quality Testing - Latest Software Testing News! ~

Want to connect with right audience?



How would you like to reach over **17,900** test professionals across **97 countries** in the world that read and religiously follow “Tea-time with Testers"?

How about reaching industry thought leaders, intelligent managers and decision makers of organizations?

At "Tea-time with Testers", we're all about making the circle bigger, so get in touch with us to see how you can get in touch with those who matter to you!

ADVERTISE WITH US

To know about our unique offerings and detailed media kit

write to us at **sales@teatimewithtesters.com**

Announcing...

Smart Tester of the Month Award !!!

❖ Prize winners:

First Prize

Name: Akash K Patel
Designation: QC Engineer
Company: Silver Touch Technologies Limited
Place: Ahmedabad

Second Prize

Name: Sonal Agarwal
Designation: Test Engineer
Company: LogicNEXT
Place: Noida

Third Prize

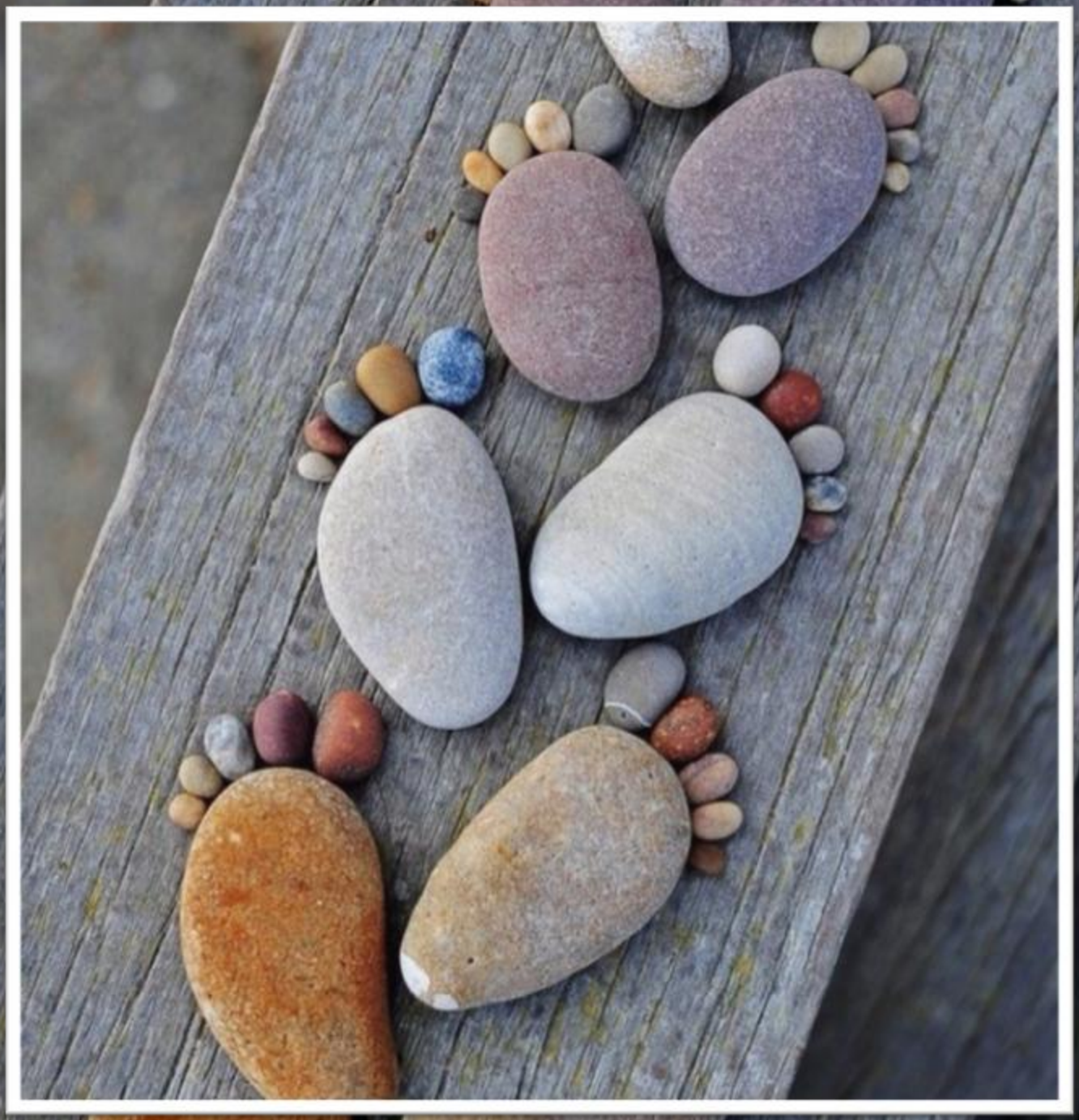
Name : Harshal Relwani

Congratulations !

Note for Prize Winners: We will inform you about your prize details via e-mail.

Names of other winners who had sent us correct answers will be declared on our Facebook Page.

Haven't you joined the gang yet? Really?



Join us on Facebook

The Fish-Eye Lens (Part 2)

Remember the Law of the Hammer?

The child who receives a hammer for Christmas will discover that everything needs pounding.

Any photographer can tell you that hammering is not likely to improve a filter, yet sometimes when I'm frustrated trying to understand a client, I start hammering away using one and only one of my filters. This is the First Law of Bad Management, or, if you like, the First Law of Bad Consulting:
If something isn't working, do more of it.

My Fish-Eye Lens, with its case full of filters, helps me remember not to be a bad consultant—at least this kind of bad consultant.

Instead, I use the First Law of Good Consulting: If something isn't working, do something else.

You may recognize this as the source of Marvin's Fourth Great Secret: Whatever the client is doing, advise some thing else.

Don's Deviance Derivation

I'm often fooled when listening to clients give "raw data" that are actually lumped in some way. Don Gause designed an exercise that taught me one way of noticing smoothed data, When speaking to a large audience, he asks them to pick "random numbers" somewhere in the range 1 to 100. When he tabulates their choices, he always finds a scarcity of numbers ending in 0 or 5. Obviously, their picks were not random at all—but influenced by their bias that "round" numbers are not random.

The members of the audience probably know that if all the figures on their tax form end in 0, the tax auditors will notice and think the numbers have been fabricated—since round numbers "can't be random." But out of millions of tax returns, some of them, at random, ought to have quite a few numbers ending in zero.

I've heard that when an expedition measured the height of Mt. Everest, they got the figure of 29,000 feet. Since this would look like an approximation, they changed the figure to 29,002 feet—less accurate, but more believable.

In other words, the world isn't (usually) that neat, so when trying to see the environment, I use Don's Deviance Derivation:

If it's too regular, it's not an observation; it's a formulation.

There's nothing wrong with smoothing the world, it's just another fact—the way people perceive their world is an essential part of the total context I'm trying to see. But I may want to go beyond—actually behind—their formulations and get to the deviant data that led them to that conclusion.

For example, Forrie, the Security Director at a client suffering from security breaches told me, "Everybody here changes their password at least once a month." The word "everybody" was a bit too smooth for me, so I asked Forrie, "How do you know this?"

He was a bit offended, then spoke me like I was a young child. "Well, I suppose you wouldn't know this, but if your password hasn't changed within the past month, it won't let you into the system until you choose a new one. Everybody here knows that."

"Oh, sorry. But you can see why I wouldn't know." That satisfied Forrie, but didn't satisfy me. Here, according to Don's Deviance Derivation, was big time lumping, and I was going to check it out.

I interviewed five people about how they handled these lockouts. One said he kept two passwords and alternated them each month. A second said he had twelve passwords —January, February, ..., December—one of which he used each month. And the other three all used the same system—when locked out, they changed their password to something simple, then immediately changed it back to the familiar value they always used.

Forrie was not pleased to receive data that contradicted his simple formulation, and he wanted me to tell him who my five informants were. I asked him why he wanted to know, and he said, "So I can get their managers to order them to really change their passwords every month!"

Forrie's formulation, it seems, went much deeper than I had originally believed. He believed that people might be messily peculiar, but not so peculiar that they couldn't be forced by mechanical means to do something that was structured, but very inconvenient.

He also believed that if they couldn't be forced mechanically, they could be forced by their managers—even though the only way their managers could possibly enforce this rule was to know their employee's passwords, thus defeating the whole point. And, finally, he believed that somehow the five people I interviewed just happened to be the only five who didn't really change their passwords every month.

All in all, Forrie taught me a lot about the context of his organization, as well as about the way Security Managers formulate the world. Of course, I suppose not all Security Managers think like Forrie. Is that possible?

Separation (or Not) of Variables

Lumping is just one way my clients become "messily peculiar" by trying to simplify their environment. Perhaps the second most popular simplification technique is just the opposite of lumping—splitting one thing into two or more relatively independent parts.

Scientists call this process "separation of variables."

To simplify the explanation let us assume that whenever we have a population of a million bacilli in a patient's lung one will have mutated to become resistant to streptomycin (S) and one to be resistant to isoniazid (I). If he is given drug S all the bacilli are killed except for the one S-resistant mutant. This multiplies despite the drug and eventually we have a new population of a million S-resistant organisms. Amongst these will be the random mutant that is also I-resistant. Now, if we change to drug I we may soon find that the patient is infected with doubly resistant bacilli. On the other hand, if we hit with both S and I together there will only be a chance of one in a million that a spontaneously occurring doubly resistant mutant will be present.

In other words, to understand how to defeat the bacilli, the scientists break down the lump called "bacilli" into two lumps, S and I. Then they hit these tiny critters with two drugs lumped together, to prevent them from separating variables. In effect, they impose the Law of Unavoidably Messy Peculiarity on the poor bacilli, which get wiped out.

Now, if I want to create a difficult problem for someone (like a hacker trying to break into a computer system), it helps to know how the brain solves problems. I can create a difficult problem by preventing a separation of variables. Against hacking, for example, I can create a set of locks, all of which must be broken at once in order to enter the system. My clients do this unconsciously, not (I hope) to make my job difficult, but to make theirs easy. But it makes my job difficult all the same, and sometimes it feels as if they are intentionally "locking" their environment from my prying eyes.

I use my Fish-Eye Lens to remind me of many tools I have for understanding the context, many of which I presented in The Secrets of Consulting. Looking at them now, it's easy to see how many of them are ways of separating variables.

For example, there's Sparks's Law of Problem Solution:

The chances of solving a problem decline the closer you get to finding out who was the cause of the problem.

Like a prism separating white light into a variety of colors, Sparks's Law helps me separate two variables the client has combined. In this case, the problem itself has been lumped with the problem of who is to be blamed for the problem.

Notice, too, that if I need to use Sparks repeatedly, I may have discovered something else about the context—that this organization is deeply immersed in the habit of blaming.

Many of the guidelines from The Secrets of Consulting are also based on separating variables. For example, there's the admonition to "Deal gently with systems that should be able to cure themselves." This reminds me to separate diagnosis from cure, being sure to do the one before even considering the other.

Another example that applies directly to understanding the context is; "Look for what you like in the present situation, and comment on it." This reminds me that clients often lump all the good aspects of the context in with the bad, and then the good get lost to their view—and to mine. By commenting,
I help them separate variables in a new way. Which brings me to those cases where the clients present me with variables already separated—possibly for their convenience, but for mine.

For instance, The Helpful Law

("No matter how it looks, everyone is trying to be helpful.") Reminds me to notice when they classify everyone as either friend or foe, a separation of variables that may not be contributing to solving their problems.

to be continued in next issue...

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.**

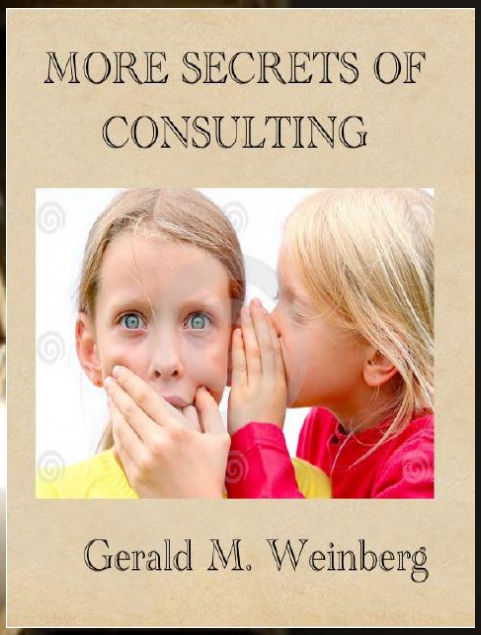
To know more about Gerald and his work, please visit his Official Website here .

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 Michael Larsen.

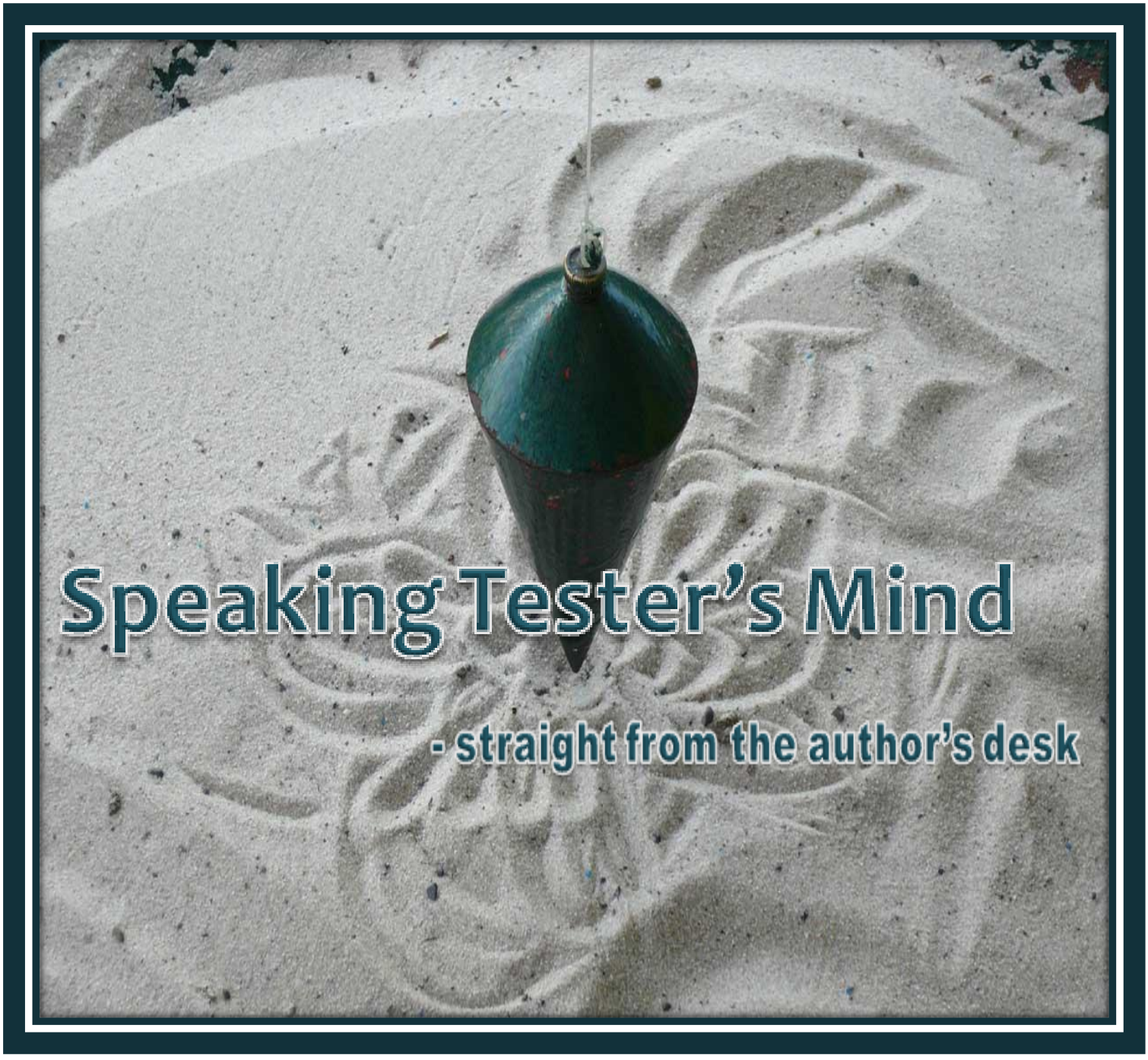


TTWT Rating: ★★★★★

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 .


Speaking Tester's Mind
- straight from the author's desk

Why is testing taking so long ?

- part 1

- By Michael Bolton

If you're a tester, you've probably been asked, "Why is testing taking so long?" Maybe you've had a ready answer; maybe you haven't. Here's a model that might help you deal with the kind of manager who asks such questions.

Let's suppose that we divide our day of testing into three sessions, each session being, on average, 90 minutes of chartered, uninterrupted testing time. That's four and a half hours of testing, which seems reasonable in an eight-hour day interrupted by meetings, planning sessions, working with programmers, debriefings, training, email, conversations, administrivia of various kinds, lunch time, and breaks.

The reason that we're testing is that we want to obtain *coverage*; that is, we want to ask and answer questions about the product and its elements to the greatest extent that we can. Asking and answering questions is the process of *test design and execution*. So let's further assume that we break each session into average two-minute micro-sessions, in which we perform some test activity that's focused on a particular testing question, or on evaluating a particular feature. That means in a 90-minute session, we can theoretically perform 45 of these little micro-sessions, which for the sake of brevity we'll informally call "tests". Of course life doesn't really work this way; a test idea might a couple of seconds to implement, or it might take all day. But I'm modeling here, making this rather gross simplification to clarify a more complex set of dynamics. (Note that if you'd like to take a *really* impoverished view of what happens in skilled testing, you could say that a "test case" takes two minutes. But I leave it to my colleague James Bach to explain why you should **reject the concept of test cases**.)

Let's further suppose that we'll find problems every now and again, which means that we have to do bug investigation and reporting. This is valuable work for the development team, but it takes time that interrupts test design and execution—the stuff that yields test coverage. Let's say that, for each bug that we find, we must spend an extra eight minutes investigating it and preparing a report. Again, this is a pretty dramatic simplification. Investigating a bug might take all day, and preparing a good report could take time on the order of hours. Some bugs (think typos and spelling errors in the UI) leap out at us and don't call for much investigation, so they'll take less than eight minutes. Even though eight minutes is probably a dramatic underestimate for investigation and reporting, let's go with that. So a test activity that *doesn't* find a problem costs us two minutes, and a test activity that *does* find a problem takes ten minutes.

Now, let's imagine one more thing: we have *perfect* testing prowess; that if there's a problem in an area that we're testing, we'll find it, and that we'll never enter a bogus report, either. Yes, this is a thought experiment.

One day we come into work, and we're given three modules to test.

The morning session is taken up with Module A, from Development Team A. These people are amazing, hyper-competent. They use test-first programming, and test-driven design. They work closely with us, the testers, to design challenging unit checks, scriptable interfaces, and log files. They use pair programming, and they review and critique each other's work in an egoless way. They refactor mercilessly, and run suites of automated checks before checking in code. They brush their teeth and floss after every meal; they're wonderful. We test their work diligently, but it's really a formality because they've been testing and we've been helping them test all along. In our 90-minute testing session, we don't find any problems. That means that we've performed 45 micro-sessions, and have therefore obtained 45 units of test coverage.

Module	Bug Investigation and Reporting (time spent on tests that find bugs)	Test Design and Execution (time spent on tests that *don't* find bugs)	Total Tests
A	0 minutes (no bugs found)	90 minutes (45 tests)	45

The first thing after lunch, we have a look at Team B's module. These people are very diligent indeed. Most organizations would be delighted to have them on board. Like Team A, they use test-first programming and TDD, they review carefully, they pair, and they collaborate with testers. But they're human. When we test their stuff, we find a bug very occasionally; let's say once per session.

The test that finds the bug takes two minutes; investigation and reporting of it takes a further eight minutes. That's ten minutes altogether. The rest of the time, we don't find any problems, so that leaves us 80 minutes in which we can run 40 tests. Let's compare that with this morning's results.

Module	Bug Investigation and Reporting (time spent on tests that find bugs)	Test Design and Execution (time spent on tests that *don't* find bugs)	Total Tests
A	0 minutes (no bugs found)	90 minutes (45 tests)	45
B	10 minutes (1 test, 1 bug)	80 minutes (40 tests)	41

After the afternoon coffee break, we move on to Team C's module. Frankly, it's a mess. Team C is made up of nice people with the best of intentions, but sadly they're not very capable. They don't work with us at all, and they don't test their stuff on their own, either. There's no pairing, no review, in Team C. To Team C, if it compiles, it's ready for the testers. The module is a dog's breakfast, and we find bugs practically everywhere. Let's say we find eight in our 90-minute session. Each test that finds a problem costs us 10 minutes, so we spent 80 minutes on those eight bugs. Every now and again, we happen to run a test that doesn't find a problem. (Hey, even dBase IV occasionally did something right.)

Our results for the day now look like this:

Module	Bug Investigation and Reporting (time spent on tests that find bugs)	Test Design and Execution (time spent on tests that *don't* find bugs)	Total Tests
A	0 minutes (no bugs found)	90 minutes (45 tests)	45
B	10 minutes (1 test, 1 bug)	80 minutes (40 tests)	41
C	80 minutes (8 tests, 8 bugs)	10 minutes (5 tests)	13

Because of all the bugs, Module C allows us to perform thirteen micro-sessions in 90 minutes. Thirteen, where with the other modules we managed 45 and 41. Because we've been investigating and reporting bugs, there are 32 micro-sessions, 32 units of coverage, that we haven't been able to obtain on this module. If we decide that we need to perform that testing (and the module's overall badness is consistent throughout), we're going to need at least three more sessions to cover it.

Alternatively, we could stop testing now, but what are the chances of a serious problem lurking in the parts of the module we haven't covered? So, the first thing to observe here is:

Lots of bugs means reduced coverage, or slower testing, or both.

There's something else that's interesting, too. If we are being measured based on the number of bugs we find (exactly the sort of measurement that will be taken by managers who don't understand testing), Team A makes us look awful—we're not finding any bugs in their stuff. Meanwhile, Team C makes us look great in the eyes of management. We're finding lots of bugs! That's good! How could that be bad?

On the other hand, if we're being measured based on the test coverage we obtain in a day (which is exactly the sort of measurement that will be taken by managers who count test cases; that is, managers who probably have an even more damaging model of testing than the managers in the last

paragraph), Team C makes us look terrible. "You're not getting enough done! You could have performed 45 test cases today on Module C, and you've only done 13!" And yet, remember that in our scenario we started with the assumption that, no matter what the module, we always find a problem if there's one there. That is, there's no difference between the testers or the testing for each of the three modules; it's solely the condition of the product that makes all the difference.
This is the first in a pair of posts. Let's see what happens tomorrow.

to be continued in next issue...

Michael Bolton has over 20 years of experience in the computer industry testing, developing, managing, and writing about software & has been teaching software testing and presenting at conferences around the world for nine years.

He is the co-author (with senior author James Bach) of Rapid Software Testing, a course that presents a methodology and mindset for testing software expertly in uncertain conditions and under extreme time pressure.

Michael can be reached through his Web site, http://www.developsense.com

Back To Index

Do YOU have IT in you what it takes to be GOOD Testing Coach?

We are looking for skilled ONLINE TRAINERS for Manual Testing, Database Testing and Automation Tools like Selenium, QTP, Loadrunner, Quality Center, JMeter and SoapUI.

TEA-TIME WITH TESTERS in association with QUALITY LEARNING is offering you this unique opportunity.

If you think that YOU are the PLAYER then send your profiles to trainers@qualitylearning.in .

Click here to know more

There was a time when people did not have compass to find right direction. The only guide they had was that guiding star up in the sky.

Do you think that you are also stuck somewhere with technical issues? Do you need help in decision making or want guidance?

Well, the wait is now over . Introducing....

"The Guiding Star"

The panel of our experts is now here to help you.
Send us your questions around software testing and our Guiding Stars will help you out.

E-mail your question on –

theguidingstar@teatimewithtesters.com

Please Note :

1. This is not a job portal.
2. Typical interview questions will not be answered.
3. Questions should be on Software Testing or related topics only.

Image courtesy : MMVI New Line Production

Do you have any Questions or Feedback on articles that we publish in Tea-time with Testers?

No Problemo! We will publish your Feedback/Comments and also the answers to your Questions that you have for our Authors.

Do write us your Feedback and Questions in below format and send it to teatimewithtesters@gmail.com :

- Your Name
- Your Brief Introduction
- Article Name
- Your Feedback or Questions if any

Make sure to write **Feedback For < Article Name>** in your subject line.


In the
school of
Testing
for your better learning & sharing experience

By Bernice Niel Ruhland

The Importance of Social Networking – Part 2

Since November 2011, I have published a series of articles on how testers can develop their skills and knowledge. I would like to take a different approach to discuss how I apply my own advice. Recently, I had to learn how to effectively test a web-based application across multiple browsers. To accomplish this task I needed to understand how other testers approach this problem and the tools they are using. I am fortunate to have a large testing network through social media and that many of these testers are bloggers.

The focus of this series does not discuss how I perform cross-browser testing. Instead it provides a wealth of information shared from the Testing Community and information from my personal research.

I would like to thank everyone who contributed information. Without their willingness to share their experiences and recommendations, I would not have been able to understand potential approaches in such a short time. A heart-felt thank you to: Ajay Balamurugadas, Mike Talks, Lisa Crispin, Martijn de Vrieze, Anne-Marie Charrett, Karen Johnson, Gagneet Singh, Dave McNulla, Moise Stedte, Akshay Thakkar, and Dorothy Graham.

Hardware

- What hardware do you need to test on?
- Do you need to test mobile phones, iPads, and tablets?
- Do you need to test on different windows configurations and macs?
- What is the cost for additional test machines?

Gather Intelligence

- Test bugs across browsers to see if it exists in all browsers. Functionality bugs may exist in all browsers where the UI bugs can be different across browsers.
- Always test bugs in the baseline browser.
- Use a defect-tracking tool to identify which browser a bug is associated. This can be a good measurement to help identify which areas of the browsers require more testing. Over time testers will learn what areas of the browsers are vulnerable and extensive testing of bugs across all browsers may be reduced.

Risk Identification

- Identify risk areas such as third-party tools, plug-ins, flash, Javascript, and Jquery. Understand from the development team vulnerable areas of browsers based upon your product. As developers check in code, have them provide cross-browser risks to the testing team.
- Only test new code or updated code against defined browsers and versions. Determine if you can make the assumption that functionality that has not been changed will not start to encounter problems.

- Over time as testers better understand the risks, update your testing approach accordingly. You may employ one approach when you initially start your cross-browser testing and modify it as you better understand the risks.

Alternative Testing Approaches

Sponsor different contests or competitions to gather initial intelligence based upon browsers and hardware. For example have a contest to test on different hardware. Encourage people in the company to bring in their machines allowing you to test on different platforms. Offer incentives such as lunch and awards for the most unusual (and oldest) machine brought in. Developers often have the best machines - your customers often do not.

If you find something unusual, ask the question "what is the customer impact, and is it worth fixing"? If you find there is a problem using an out-of-date Netscape browser on NT - are you just going to log this, and not fix it? Before submitting any bugs to your tracking system, have someone review them to esure they are worthy to be fixed. You can create your own type of contest that meets the needs of your project and what the employees will enjoy participating.

Test Lab

- Testers can be using different versions of IE. It can be difficult to control IE versions since updates tend to download the most current version.
- Determine if you should download the new version and keep it in a safe location for future testing needs. This allows you to load any version of a browser to the tester's machine.
- Will the testers have a virtual machine with different browsers loaded? Consider the cost of licensing the operating system for each machine. Or is it better to have multiple workstations to support the various browser versions? Consider the opportunity of sharing workstations across testers.

Mobile Devices

- The demand of mobile phone usage is increasing over the years. Understanding the consumer demand to use your product on a mobile device is important. Additional information on this can be found at: http://royal.pingdom.com/2011/12/07/the-mobile-web-in-numbers/
- Another article to read: Browser Choice: A thing of the past? http://news.cnet.com/8301-1023_3-57439936-93/browser-choice-a-thing-of-the-past/

How do I know how much time to allocate to testing other versions?

- Conduct a browser test through session charters. A tester and developer work together through a few charters to identify how long it takes. With this information, start to build time estimates.
- Ideally more time is allocated initially to gather more intelligence about bugs across browsers and versions.
- Identify functionality that needs to work as part of Compatibility Testing for defined browsers and versions. Again, not all browser versions will be tested.
- Add a day for each new feature to test basic functionality through performing exploratory testing.
- If more extensive testing is required, add 50% of the original testing time to test an additional browser. Testing time should be reduced basd upon what was learned through the initial testing.
- Use the minimal subset of testing to understand how long that testing typically takes and multiply across number of browsers.

Back To Index

Bernice Niel Ruhland is a Software Testing Manager for ValueCentric, LLC a software development company located in Orchard Park, New York. She has more than 20-years experience in testing strategies and execution; performing data validation; and financial programming.

To complete her Masters in Strategic Leadership, she conducted a peer-review research project on career development and on-boarding strategies. She uses social media to connect with other testers to learn more about their testing approaches to challenge her own testing skills.

The opinions of this article are her own and not reflective of the company she is employed with. If you have any questions / comments on this article or if you would like to connect, Bernice can be reached at:

LinkedIn: http://www.linkedin.com/in/bernicenielruhland

Twitter: bruhland2000 G+ and Facebook: Bernice Niel Ruhland

Software Performance Engineering :

"WORKLOAD ANALYSIS" – KYC: Part 1

by Samajeet Mohanty

The objective of this article is to familiarize reader with the concept of something called "***Workload***" and how to analyze it to formulate meaningful performance test scenarios. It is vital for a performance engineer to understand the workload behaviour as it can help in non-trivial exercises like ***Capacity Planning or Hardware Sizing*** for future and model the architecture of SUU accordingly. Workload Analysis is also referred to as ***Workload Modelling***.

If you are wondering what **KYC** stands for in article's title, it is 'Know Your Chicken' as I was enjoying it while writing this paper. Nah!!…just kidding. I would like to phrase it as ***Know Your Customer*****,** in this context. Stay with me and as we go along through rest of the article you will understand why I mentioned it in here.

Alrighty then!!....Let's begin. If you've gone through the 2nd article in SPE series, I hope you remember the 2nd phase in PELC (see glossary at end for definition of PELC) where performance engineer has the responsibility of talking to Business Analyst and, if required, Post-Production Monitoring team to gather few end-user related statistics. The sample spreadsheet below is a template of what possible questions arise in this phase that would help PE to model appropriate and realistic tests scenarios.

User Type	Transaction Type	Normal & Peak Season (1 day stats)			Total Application Data Size in DB (I)	G + 1 Yr	I + 1 Yr
		# Average Concurrent User Load	# Peak Concurrent User Load (G)	Duration of Peak Load			
Buyer	Product Search	300	800	1 hour	10 GB	3000	100GB
	Product Comparison	150	300	30 min		500	
	Product Booking	300	600	1 hour		2000	
Seller	Product Upload	5	20	30 min		50	

The Workload pattern generally differs to a great extent during normal season than a holiday season. Also, please keep in mind that the AUT is being tested to ensure it survives not only the current load but also the load 1 or 2 years down the line.

It should be analyzed –

- Who are the different users accessing AUT and their roles.
- What geographies are they accessing it from.
- What actions are being performed on AUT by those users and at what frequency.
- What background jobs / processes are running in parallel when these users are on the system and are those jobs manual or automatic.

Some of the other parameters worth enquiring are:

Parameter Type	Sample Description with Example
Transaction Mix	Which transactions are expected to occur in parallel ? Eg: Few Customers might be searching for a product while others might be buying them
Transaction Rate	Product booked / Hour ? , Items Searched / hour ?
Transaction Workflow	Which app and Infra components does, say "Product Search", transaction touch from End-2-End ?
Interaction # per transaction	How many requests does application make to the database, say for product booking by 1 customer ?
Communication Protocol between various app and infrastructure components	How does information flow between Application and Database server - HTTP / FTP, DB Query ?
Data Traffic and Type	How many and What type of data is flowing per screen per request - JPEG / JSP / Javascript ?
Transaction Complexity	How many dynamic elements are being fetched per transaction per request ?
Online & Offline Batch Processing - Volume , Schedule	What, When and How much of batch processing happens ?
Any migration / DB backups planned	When are they planned ? , Any online / offline activity expected during that period ? , What Performance tests need to be redone to verify the smoothe operation of new environment ?
Inbound and Outbound messaging interfaces for AUT	Is AUT a producer or consumer of JMS messages and to/from whom ?

As seen in Figure above, IOzone issues different types of IO operations like read, write, parallel read & write so on against the disk array of VMAX and calculates the IO throughput (IOPS) based on the results obtained from the storage component. Few other tools to test storage component performance are Bonnie++, Orion and Iometer.

Most of the time the parameters mentioned above would only be the best estimate of the Subject Matter Experts for any new application being deployed. However for an existing application which is being upgraded / migrated / re-architecture'ed, monitoring the network connections, Web Server and application server logs can provide vital information like number of concurrent user sessions throughout a day/month, their transaction activities, process dependencies and so on. These help the testing team to simulate conditions which do not Overload the system and at the same time not Under-Utilize it too.

Now that we know the workload pattern, it's time to move on to the second half of KYC, which is Transaction Processing Analysis. I'll cover this in one of my upcoming chapters.

APPENDIX:

KYC → Know Your Customer

SUU → System Under Use

PELC → Software Performance Engineering Life Cycle

KPI → Key Performance Indicator

HS → Hardware Sizing

SUT → System Under Test

OLTP → Online Transaction Processing

PE → Performance Engineer

Back To Index

Samarjeet Mohanty (Samar) is a, self-proclaimed, practitioner of anything to do with Software Performance Engineering world.

He's quite experienced in Performance Engineering and Testing methodologies of software applications (BFSI) using industry standard tools and techniques.

Apart from being an active participant in technical forum's concerning performance and generic testing QA, Samar likes to interact with creative-minded professionals from all walks of life to better understand their take in the related field.

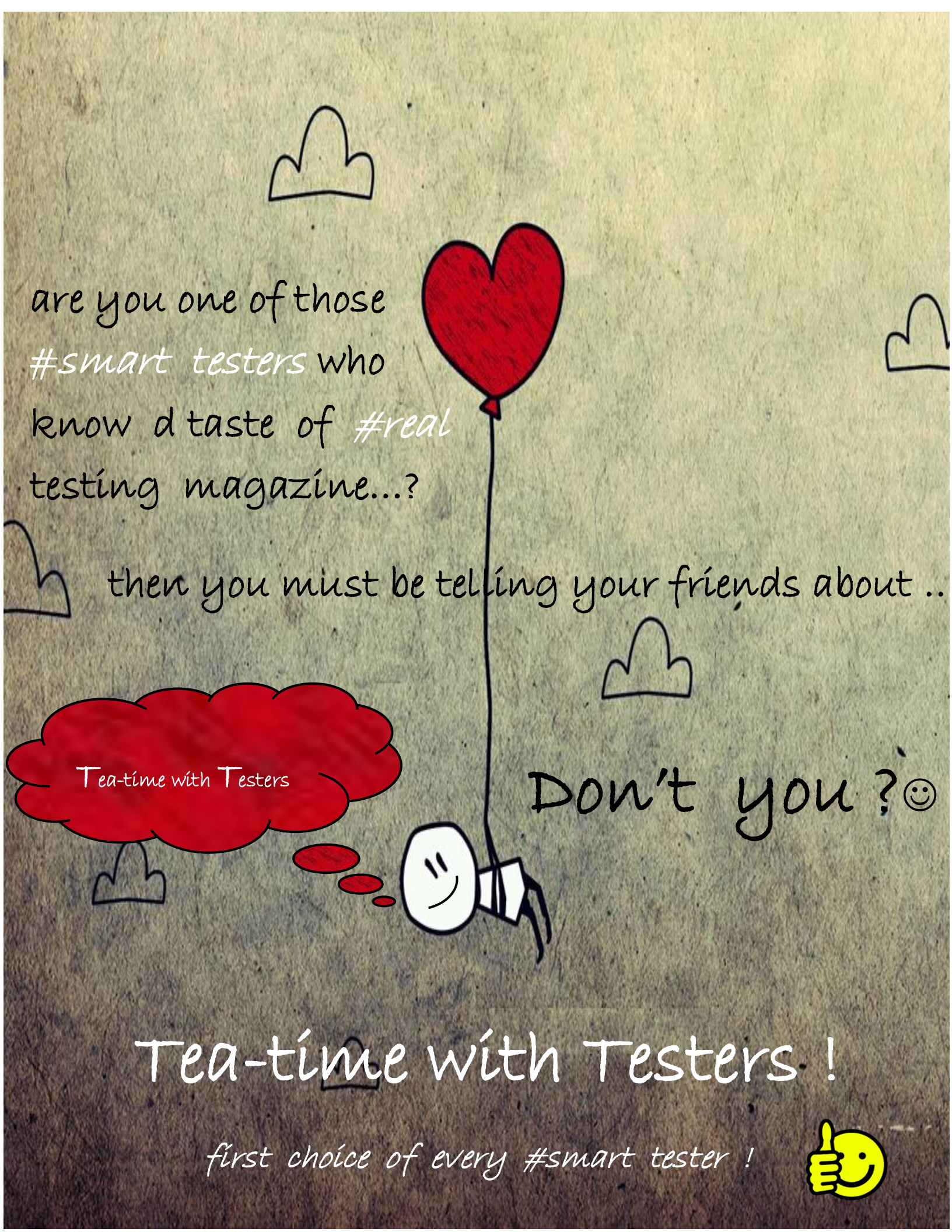
If interested, connect on:

LinkedIn:

http://ca.linkedin.com/in/samarjeetm

Twitter:

http://twitter.com/SamarjeetM


are you one of those #smart testers who know d taste of #real testing magazine...?
then you must be telling your friends about ..
Tea-time with Testers
Don't you ?☺
Tea-time with Testers !
first choice of every #smart tester !

testing intelligence

- its all about becoming an intelligent tester

an exclusive series by **Joel Montvelisky**

My 5 tips when working with distributed testing teams

Not too long ago, when you said you were part of a global software development team it meant that you were working for IBM, SAP or one of only a number of Fortune 100 companies that had development centers around the world.

Today, even companies with less than 10 employees can have teams distributed in two, five or more geographical places. The location of team members has become an accidental attribute, often disregarded or altogether ignored.

If you add to this alternative testing or development sources such as Offshoring, Outsourcing or even (one of my favorites) Crowdsourcing, it becomes apparent that only a small percentage of companies today are still doing 100% in-house development and testing, like we used to do in the "good old days".

Managing distributed teams was and still is challenging.

About 13 years ago I was introduced to the concept of offshore management. As a young QA Manager I remember how excited I was when my boss told me that I would start managing, in addition to my local team of testers, a group of 6 test engineers in another country. This was an experiment on offshoring (actually it was outsourcing) that my company wanted to do, and me and my team were chosen as the company guinea pigs.

I only thought of how much more I would be able to do now that I had a bigger team, and I completely failed to realize that this happiness came with a significant price tag. Within a couple of weeks I understood that managing a remote team was at least 10 times harder than managing engineers sitting next to me.

My main challenges (mistakes?) came from not understanding how my extended team actually thought and worked. The cultural differences, together with the difficulties in communication, made it close to impossible to get visibility into what they did and how they did it.

To cut to the chase, after 6 months of struggling I asked my manager to close down the experiment, as it was taking me more time to manage them than I was gaining from their overall work.

5 tips for succeeding with distributed teams

Throughout the years I've had plenty of chances to redeem myself from my first failed attempt at outsourcing, and I've worked out a list of tips that today help me approach these project correctly and succeed in leveraging the potential of my distributed development and testing teams.

Here are my 5 tips to succeed when embarking in any distributed team project:

Tip No 1
Ensure communication redundancies

When working on-premises, each of us has plenty of ways to communicate with the other team members. We can talk during regular meetings, jump to the cube of a peer, send him an email, leave a hand written note on his desk, or even go out to coffee or lunch and talk about stuff we prefer not to talk during "regular work hours".

I remember working on a team in particular where the important decisions and the revolutionary ideas were reviewed and closed "informally" while smoking in the parking lot at 8:00 PM after the end of the working day…

This is simply not the possible when you work with distributed teams.

The point is that since not all messages can be convene through the same channel it is important to make sure you have in place multiple channels of communication, and that these channels are available to communicate with all team members (regardless of their jobs or ranks).

Some practical ideas:

- Create a list of all team members where you have all the ways to contact each person including office phone number, mobile number, mail, Skype, etc. Add some personal info as well, starting with a picture of the person, date of birth, hobbies, marital status, etc.

- As a manager make it a habit to start your day by greeting your off-site team members via Skype, just like you do with your on-site team when you make your morning coffee or walk by their cubicles.

- Encourage your team to use the phone or Skype instead of emails to ask quick one-to-one questions. A big mistake done by many people today is to use long email chains (that expand over hours or days) instead of picking up the phone and solving the issue in 2 to 5 minutes.

- Video conference as much as possible, body language is as important (or more) than verbal language. With Skype, GoToMeeting, FaceTime and all the rest of the communication systems it is easier and cheaper than picking up the phone.

Tip No 2

Make sure everyone knows what everyone else is working on

Take 30 minutes each morning (or afternoon, depending on where the other teams are) to have a synchronization meeting where each team member will say what he did yesterday and what is his task for the day. Make sure everyone knows what everyone else is working on and encourage engineers to provide feedback and help other team members, regardless if they are in the same site or not.

If you are working on an Agile environment this will probably be your standard "Daily", but most teams don't work this way.

Still, regardless if you are working Agile, Waterfall or using your own methodology, this type of synchronization meeting will allow your individual team members to feel they are involved and are part of each other's work.

It will also allow you as manager to know what each team member is doing and what are his challenges, without needing to micro-manage their work.

One thing to take into account is to keep these meetings short and focused. If they become endless and boring discussions between specific members of the team or updates where no one else is paying attention to what the others are saying then they will become more harmful than helpful…

Tip No 3

There is no replacement for personal contact

There's no way around it, the best way to get to know someone is by meeting him in person and spending some time together.

If you've been placed in charge of a remote team, schedule a trip and go to meet them in person as soon as possible. It will show them you care and are interested in the team and what they are doing.

Make sure personal meetings are not limited to managers. Put in place an exchange program where engineers from all sites go and work with other teams for 2 or 3 weeks once a year.

This will allow people to get to know each other personally, and will help to share work dynamics, methodologies and even the challenges each team experiences as part of their day-to-day operation.

Tip No 4

Manage your work on a single platform

If you want your team to work together they need to "talk the same work language" – and I don't mean switching to English!!!

Centralize their work on a single management platform, this will give them a "shared language" to communicate and work.

Switching to a unified platform is not trivial and there are number of things you should to take into account along this process:

- The platform should be accessible and comfortable to all members on all sites. Make sure that things like accessibility and response time will not make it frustrating to work for testers on remote locations.
- The idea of the platform is for all teams to work in a similar fashion, so it should accommodate the methodology of all teams and be customizable to needs of all.
- Organize the information in ways that will let all engineers access the information from all the teams and be able to leverage it for their own work.

Once you are working on a single platform find ways to encourage people to access and use the information from the other teams. This will help team to relate to one another and will trigger a direct feedback process between the teams.

Tip No 5

Make sure the "away" team is 100% identified with the project/product

When you have a number of teams working together there is always one (or more) that will feel "away" or "disconnected".

There are many reasons for this, maybe because only one of the testing teams is working on the same site as the development team and they are "closer" to the developers, or because one of the teams sits in the site of corporate headquarters, or simply because one team is bigger and "older" than the other.

Whatever the reason this can cause the feeling of one team that is closer and the other further away from the "action".

There are a number of things you can do to work on these feelings and improve the attitude of the other team, for example:

- Make sure to let the testers of the "away" team have direct access to the developers they are working with. Do this via Skype, video conferences, etc.

- Set weekly update video meetings with people from the company that are not related to the development such as marketing, finance, sales, etc. where they talk about how other teams work, what are their challenges, etc.

Whatever the issue, something that always works for me is "merchandising"! YES, as simple as giving them corporate notebooks and pens, making sure they have corporate posters on their walls, giving away t-shirts or stickers or any other type of brand or corporate merchandise has a very positive effect on the way your "remote" team will identify with the Company.

Working on the psychological and physical level

To summarize, if you want to succeed on your distributed project you need to remember to bridge over the psychological as well as the physical barriers affecting your collaboration and communication.

Treating only one of these aspects while leaving the other one unattended will ultimately cause you to fail.

Do you have other ideas and tips that can help to succeed on distributed projects? Please share them with us!

Back To Index

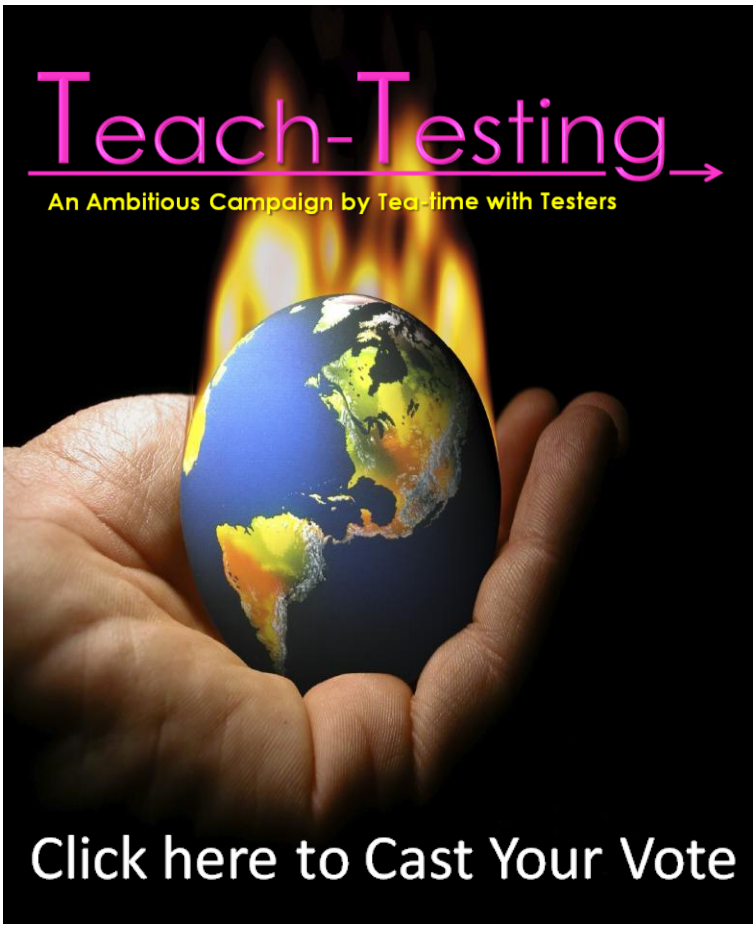
Joel Montvelisky is a tester and test manager with over 14 years of experience in the field.

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




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...!
Click HERE to read our Article Submission FAQs !
sciencephotogallery

Mirror, mirror on the wall, who is the fairest of all?

Snow white and the seven dwarfs is a classic story we all grew up with. The story illustrates many things, in addition to the power of love.

I have been inspired by the mirror in the story. The mirror that reflects how good you look, every day. As technical folks wanting to deliver the highest quality, we do very many things via better techniques, tools and process. How do we know if these are yielding results? Reflect. Use a mirror. A dashboard containing measurements that matter, that is real time, that helps us adapt constantly and change for the better.

Recently I had interesting conversations with senior QA folks from non-software domains. The gentleman from the fashion and apparel industry said "We are a labour intensive industry, and a lot of variables/factors like fabric, thread, peoples' mood, equipment etc affect the quality of the apparel we make. And customers don't tolerate poor quality. Know what we do? We implement continuous feedback via a real-time dashboard." Aha - the mirror!

I started thinking that we do; we collect measurements related to quality, progress and present them as reports/charts. Most often these are intended for managerial decision making. The person from the high

tech manufacturing domain seemed to read my mind, he pitched in and said "It is not about creating great dashboards to help the manager, it is about empowering the people on the shop floor to assess the situation and make on-the-spot decisions to course correct rapidly. It is about changing behaviour".

The fashion and apparel gentleman added "It is not just collecting measurements related to defect counts, it is knowing about kinds of defects produced and their distribution over time". That is when I related to HBT, where the focus is on defect types.

The spectacled gentleman from the high tech automotive industry added "Note that it is great to know about defect types and distributions, but in our industry, we sensitize ourselves deeply to prevent them, as the cost of fix is very high in our industry. We do quite a few things like modeling, simulation to ensure 'wellness', not just 'treat well'"

Hmmm...it is not just enough to see the issues in the mirror and work on them, it is not just validation, it is about embedding quality in. A mirror that changes me. Profound.

The manufacturing gentleman added "We develop components for high technology industries; the key is to understand the context of usage and look for defects. Testing a component alone is insufficient; we need to visualize the usage context".

The bulb lit up... The background needs to be reflected too, not just my work. Wow.

The lone gentleman from the software (mobile) commented "We do not always need the "ultimate quality"; in certain situations we can tolerate some deficiencies". Not wanting to offend the others he clarified "This is the concept of 'technical debt'. The tolerance to deficiencies is simply not there in mission critical industries, where product shelf life is long. Whereas in our industry (mobiles -fast moving products) shelf life is low, hence there is tolerance for some deficiencies".

Hmmm... This implies that sometimes 'my' reflection is not as perfect, but heck it is ok in certain situations.

After the conversation, I reflected. All we need is just a mirror to do better. Reflect work outcomes, along with the background (usage context) and tolerance of the image for that context! This changes you. It changes your approach/behaviour to quality and therefore how and when you apply.

Find your mirror. Those set of measurements that help you understand the quality of your work and the quality of the product. Those that can change you. Change you do deliver excellence. The perfect reflection.

Mirror, mirror on the wall, who is the fairest of all? I always want it to be me i.e. my product.

Have a great day.

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** .

Back To Index

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.

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, Smartphones 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.

by

Quality is delighting customers
QUALITY TESTING

Testing PUZZLES by Sebi

Claim your Smart Tester of The Month Award. Send us an answer for the Puzzle and Crossword bellow b4 18th August 2012 & grab your Title.

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

Exciting PRIZE for 1st three WINNERS*

NOTE : S.T.O.M. contest comprises of Testing Puzzle + Crossword. To claim their prize, participants should to send answers both for puzzle and crossword.

*CONDITIONS APPLY.

“Solve This Puzzle”

Biography

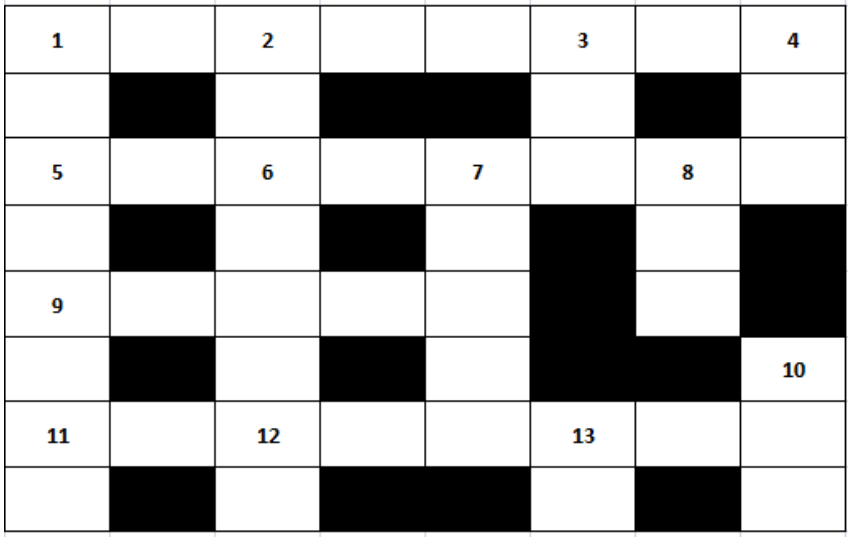
Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years. He is currently hosting European Weekend Testing.

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

Back To Index



Horizontal:

1. Which software update rendered many PCs Useless? (8)

5. It is a tool that executes plain-text functional descriptions as automated tests (8)

9. A measure of the probability and severity of undesired effects, is called _____ (4)

11. A document describing the estimation of the test efforts, approach, required resources and schedule of intended testing activities, is called ____(8)

Vertical:

1. Testing is used to confirm the design and /or performance of security controls implemented within a system. It is called _____ testing (8)

2. It is a series of graphical user interface-based operating systems developed by Apple Inc. (3)

3. A chronological record of all relevant details about the execution of a test. It is called _____, in a short form (2)

4. Testing of programs or procedures used to convert data from existing systems for use in replacement systems, in short form (2)

6. The short form of Computer-Aided Software Testing (4)

7. It is a free web functional testing tool (4)

8. It is a black-box GUI test automation tool. Its first 3 words (3)

10. Short form of Multinational Company (3)

12. Testing that attempts to discover defects that are properties of the entire system rather than of its individual components. In short form (2)

13. It is the testing of a resource or resources multiple times under program control. It is called _____, in short form (2)

Answers for last month's Crossword:

T	E	S	T	I	T	E	M
E	■	O	■	T	■	T	■
S	O	A	K	G	R	A	Y
T	■	P	■	■	T	■	S
P	O	U	N	D	E	R	M
L	■	I	■	R	■	■	O
A	■	■	■	E	■	■	K
N	E	G	A	T	I	V	E

We appreciate that you

"LIKE" US !

Join us on Facebook.

You are just a CLICK AWAY


Every Tester
who reads Tea-time with Testers,
Recommends it to friends and colleagues .
What About You ?
Image : vernhart

Our Testimonials

Great job team. This magazine stands out in a crowd.

- Kavitha M.

I used to think that magazines are just to enhance your knowledge but TTWT is kind of magazine which serves as practical guide. I admit that your articles are insightful and powerful enough which have forced me to change my thinking and beliefs about testing. Now, I will surely think beyond *Test Plan* and *Pass/Fail Count* ☺.

Many Thanks.

- Diana

Great content. I loved the articles by Guru Jerry, T Ashok and Joel.

Can't wait for the next issue.

Regards,
Tim

in next issue

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)


Lalitkumar


Pratikkumar

Contribution and Guidance:

Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)


Jerry


T Ashok


Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Image Credits- Indulgy / DK Photography

Core Team:

Anurag Khode (Nagpur, India)

Dr.Meeta Prakash (Bangalore, India)


Anurag


Dr. Meeta Prakash

Testing Puzzle & Online Collaboration:

Eusebiu Blindu (Brno , Czech Republic)

Shweta Daiv (Mumbai, India)


Eusebiu


Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)


Kiran Kumar


Chris


Romil

|| Karmanye vadhikaraste ma phaleshu kadachna | 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