

THE **INTERNATIONAL** MONTHLY ON SOFTWARE TESTING

# Tea-time with Testers

**Jerry Weinberg**

Observing and Reasoning about Errors ( Part 2)

**Matt Heusser**

Professionalism in the World of Software Testing

**T Ashok**

The mistaken notion of Structural Testing

**Selena Delesie**

Learning for Software Testers

**David Burns**

Getting Started with  
Selenium webdriver and Python

**Olaf Lewitz**

Balance : Teaching Vs Coaching

**Ola Hylten**

Communicate?

I am communicating ! They just don't listen !

**Joel Montvelisky**

Testing Manual and Automation together !

**Anurag Khode**

Testing 3G or Wi-Fi Connection Speed!



EACH ISSUE IS  
A **GREEN** ISSUE

September 2011 | Year 1 Issue VIII | [www.teatimewithtesters.com](http://www.teatimewithtesters.com)

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

THE **INTERNATIONAL** MONTHLY ON SOFTWARE TESTING

## Tea-time with Testers

**Jerry Weinberg**

Observing and Reasoning about Errors ( Part 2)

**Matt Heusser**

Professionalism in the World of Software Testing

**T. Ashok**

Structural Testing

**Selena Delesie**

Learning for Software Testers

**David Burns**

Getting Started with Selenium WebDriver and Python

**Olaf Lewitz**

Balance: Teaching Vs Coaching

**Ola Hylten**

Communicate? I am communicating! They just don't listen!

**Joel Montvelisky**

Testing Manual and Automation together!

**Anurag Khode**

Testing 3G or Wi-Fi Connection Speed!



EACH ISSUE IS  
A **NEW** ISSUE

September 2011 | Year 1 Issue VIII | [www.teatimewithtesters.com](http://www.teatimewithtesters.com)

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

Dear Readers,

I am pleased to offer you the 8<sup>th</sup> issue of **Tea-time with Testers** with yet another feast of brilliant articles...No! Delicious articles!

Well, I have got something; to share with you this time. This September month had a lot more in store to shower on me. Series of events occurred which taught me many good things and what I can conclude in the end is, "**professionalism in the field does matter.**"

Please read **Matt Heusser's** article on *professionalism in the field* and you'll come to know why I said so. I thank Matt for his valuable contribution in this issue.

Another thing that I got to learn is:

**"Doesn't matter how much you know; your learning is never enough."**

If that makes you curious to know *what and how testers should learn*, then please read **Selena Delesie's** article *Learning for Software Testers* and please do not forget to thank her for it.

**Jerry's** writings will always guide us for ages to come and once again; I thank him for contributing in Tea-time with Testers.

I also got to learn that:


**"Skills are to be improved and not just to be possessed".**

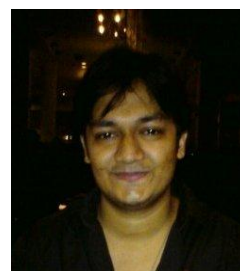
If you desire to improve on your testing skills, then we have **a treasure of legends** to help you. Check out the articles by **T Ashok**, **Joel Montvelisky**, **David Burns**, **Olaf Lewitz**, **Ola Hylten** and **Anurag Khode**. You will love them all for sure.

Well, that is all friends for this time.

Oh Yes! We have some surprise for you in next issue. Till then Enjoy Reading and have a 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](mailto:teatimewithtesters@gmail.com)

Pratik: (+91) 9819013139

Lalit: (+91) 9960556841

© Copyright 2011. Tea-time with Testers.

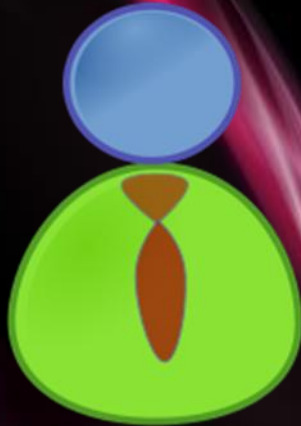
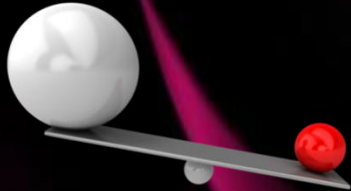
This ezine is edited, designed and published by **Tea-time with Testers**.

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



## Testing Puzzles

by Sebi



Crossword  
by



## Editorial

### What's making News?

### Tea & Testing with Jerry Weinberg

### Speaking Tester's Mind

Communicate? I am communicating! They just don't listen! -16

Balance: Teaching Vs Coaching - 21

Professionalism in the world of Software Testing -25

### In the School of Testing

Learning for Software Testers - 32

Getting Started with Selenium and Python- 39

Testing 3G & Wi-Fi Connection Speed - 40

Testing Intelligence: Testing Manual & Automation Tests together are challenging - 45

### T' Talks

The mistaken notion of Structural Testing- 50

### Tool Watch

### Testing Puzzle – S.T.O.M. Contest

### Our Testimonials

### Family de Tea-time with Testers



image: [www.bigfoto.com](http://www.bigfoto.com)

## QAI kick starts nationwide road show on careers in software testing

India Infoline News Service , Sep 15, 2011

**The event was addressed by Pradeep Chennavajjula, CEO Edista Testing Institute, a wholly owned subsidiary of QAI with an interactive session with the students on the trends in the IT industry and software testing as a career.**



QAI the leading workforce development and consulting organization rolled out a nationwide road show campaign in software testing targeted at students entering the professional world. The ten city road show campaign started off in Allahabad. The event received overwhelming response with over 400 students from leading engineering and technical colleges from in and around the city of Allahabad. The event was addressed by Pradeep Chennavajjula CEO Edista Testing Institute, a wholly owned subsidiary of QAI with an interactive session with the students on the trends in the IT industry and software testing as a career.

Commenting on the initiative, Pradeep Chennavajjula, CEO, Edista Testing Institute, "As the Software testing market is growing rapidly with niche avenues, students can plunge into functional testing or automation testing. QAI will address these requirements and be at the forefront of providing suitable offerings for workforce development across



the country in various business critical skill domains through a portfolio of certifications, training processes and assessments. "

He further elaborated, "According to the latest Gartner report, worldwide software testing market is expected to be at \$13 billion and the global market for outsourced testing to be around \$6.1 billion of which India is expected to contribute a 70% share. Opportunities for software testers are enormous. Presently the industry employs about 1,30,000 people and this number will grow enormously considering the exponential demand in the industry. In fact, these days software testers are paid more than software developers."

The daylong event demystified the perceptions and myths around Software Testing and discussed in detail the role of software tester in the industry today. Further, discussions around soft and hard skills required by software testers were outlined along with the roles and responsibilities, career path and benefits were discussed in detail. Software testing today is a viable career option for students and a lucrative career opportunity for students passing out from engineering and technical colleges.

## Selenium 2 Release: Behind the Scenes



By David Burns

On Sept 20 , 2011

---

A Selenium 2 has recently been released. With this release my fellow Selenium Committers are really excited to be releasing one of the best browser automation frameworks in the world.

For those who don't know what Selenium is let me explain. Selenium is a framework that allows developers and testers to create automated tests against a web application by controlling the browser. It was created by Jason Huggins while working at ThoughtWorks. It was created as a tool to make sure that the quality of the ThoughtWorks Time and Expense Reporting application was high. Jason and the team he was on went from breaking the application in Internet Explorer to breaking in Firefox.

A few years later, Simon Stewart created a different automation framework for driving the browser. He called his project Webdriver. He wanted to solve the issue of driving the browser but from a different direction. Simon was trying to solve slightly different issues while driving the browser.

But these two projects happily co-existed. Then Simon and Jason had a discussion about merging the projects.

This lead to the creation of Selenium 2. Each project had their strengths and weaknesses. Fortunately where one had a weakness the other had strength. A number of people have hit issues with Selenium in the past, for example using a unsecure site and trying to log in, which redirects you to the secure version of the site. Because Selenium is written in JavaScript the browser would block that type of action. The reason for this is Same Origin Policy in browsers. It prevents JavaScript coming from one place interacting with pages that are not from the same place. This is a standard security feature in browsers.

The other issue commonly hit by Selenium users is that they would like to do a file upload. Again, Selenium can't do this due browser security restrictions. JavaScript can't access file input (<input type=file>). Browsers prevent it from accessing that type of element because it could mean that a

password file could be uploaded without your knowledge. These problems are easily solved by WebDriver however WebDriver didn't support as many browsers as Selenium but in the last few months we are seeing more support being added. We have also seen that WebDriver supports mobile devices, like iOS and Android, so that you can automate your site on your iPad. It works both on the device as well as in the emulator. As we all know, Mobile is going to be the future with the rise of smart phones and tablet devices so we have to put a lot of effort into making sites work on these devices.

So now with the merger between the two projects we can test web applications in a way that was never really possible. We can draw on canvas elements as shown with this tutorial (<http://www.theautomatedtester.co.uk/blog/2011/selenium-advanced-user-interactions.html>). This is something that people have wanted to do for a while now. There are a number of different problems that have risen from working with HTML5 but to be honest if it wasn't challenging we wouldn't be in this business.

So go have a play with the new WebDriver API and see how much more you can do with your browsers. If you want to see some examples of how Mozilla do their Selenium tests, have a look at Mozilla Organization on Github (<http://github.com/mozilla/>).

If you would like to see what my team, Automation Services (<https://lists.mozilla.org/listinfo/dev-automation>) and the web QA team (<https://mail.mozilla.org/listinfo/mozwebqa>) are up to sign up to their mailing list. There will be discussions on what and how we are doing this as well as discussing test days so you can help get involved!



For more updates on Software Testing, visit → [Quality Testing - Latest Software Testing News!](#)

Software Testing NEWS

Announcing...

## Smart Tester of the Month Awards !!!

### ❖ Winners for Testing Crossword :

Ms. Dharshini .S, Congruent Info Tech India

Mr. Nishant Thakur , Adobe Systems India

Mr. Sudhamshu Ailineni , S2 Technical Strategies

Ms. Sudharani Devarajan , Cognizant Technology Solutions

# Congratulations !



Darshini S



Nishant Kumar Thakur



Sudharani Devarajan



Sudhamshu Ailineni



# Look Who's Rising



Eight Months

5000+ Readers

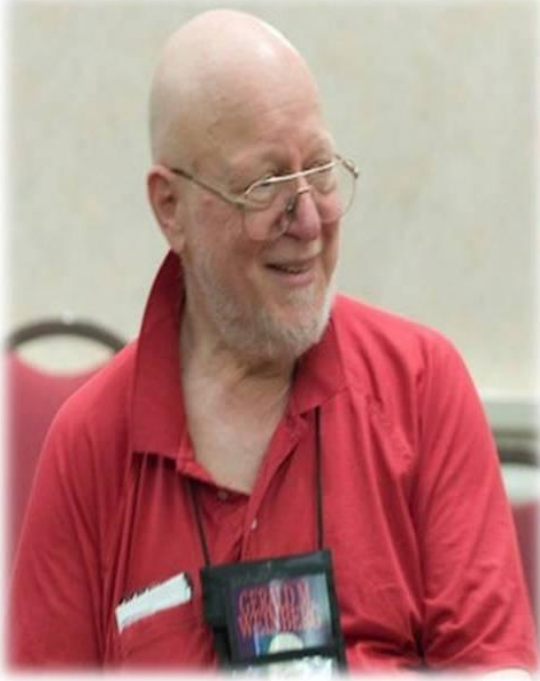
74 Countries

ONE Magazine

## TEA-TIME WITH TESTERS

Subscribe [here](#) Right Away to get our all Issues for FREE

# Tea & Testing



with

Jerry Weinberg

## Observing and Reasoning About Errors (Part 2)

### 1.2. Mis-classification of Error-Handling Processes

By the term "error-handling process," we'll refer to the overall pattern that has to do with errors, a pattern which can be resolved into several activities. Once we understand the distinction among these component activities, we'll be able to describe the dynamics of each in a way that will suggest improvement.

Characteristically, however, Pattern 1 and Pattern 2 organizations are not very adept at knowing just precisely what their error-handling process is. If you ask, the typical answer will be "debugging." With that sort of imprecise speech, improvement in fault-handling is unlikely.





### 1.2.1. Detection

Detection of faults is achieved in different ways in different software cultures. Pattern 1 and 2 organizations tend to depend on faults being detected by failures in some sort of machine execution of code, such as machine software testing, beta testing, and operational use by customers. These are the STIs.

Pattern 3 organizations also detect faults through failures but tend to prefer going directly to faults by some process that does not require machine execution of the code. These mechanisms include accident (such as running into an error while looking in code for something else), technical reviews of great variety, and tools that process code, designs, and requirements as analyzable documents, suggesting failures without machine execution of the code itself. These methods result in SFA which don't necessarily correspond to any STI, if they were applied early enough to prevent any failure resulting from the fault.

### 1.2.2. Location

Location, or isolation, is the process of matching failures with faults. Even when a fault is found directly, as in a code review, good practice dictates that the SFA contain a trace forward into the set of failures known to exist. Only by forward tracing can unsolved failures be cleared out of the STI data base. If a great many unsolved STIs remain, managers and programmers have a tendency to discount all of them, which makes location of truly active STIs more difficult.

### 1.2.3. Resolution

Resolution is the process that ensures that a fault no longer exists, or that a failure will never occur again. A failure may be solved without having its fault or faults removed. Removal of faults is an optional process, but resolution is not. Resolution of an STI may be performed in several ways:

1. Remove the fault that led to the STI. This is the "classic" way of "debugging."
2. Define the STI as unimportant, such as "too minor to fix," or "non-reproducible."
3. Define the STI as not arising from a fault in the system, but usually as a fault in the person who reported it.
4. Define the fault as not a fault, such as by following the Bolden Rule that says, "If you can't fix it, feature it."

In troubled Pattern 2 organizations, the majority of STIs are resolved by 2, 3, and 4, while management believes they are resolved by (1).

### 1.2.4. Prevention

Prevention may seem a pie-in-the sky approach to people buried deep in Pattern 1 and Pattern 2 organizations. The history of other engineering disciplines assures us that some schemes for preventing errors will ultimately prevail, but these seem a long way from where most of my clients are standing today. When I show them articles about Pattern 3 organizations, they say they're not applicable to their organizations. When I show them articles about "clean room" software development or other Pattern 4 techniques, they simply chuckle with disbelief.



In fact, however, most of the error work in a software development organization is actually prevention work, though Pattern 2 managers don't understand this. Only after they become rather sophisticated in analyzing software engineering dynamics do they realize that most of their activities are in place to prevent error, not fix it. Just to take one example, ask people why they follow the practice of "design before code." Very few of them will recognize this rule as an error-prevention strategy dictated by the war against the Size/Complexity Dynamic.

### 1.2.5. Distribution

In Pattern 2 organizations, distribution of errors is an important and often time-consuming activity. By distribution, we mean any activity that serves to prevent attributing errors to one part of the organization by using the technique moving them to another place. Here are some examples of distribution:

1. Developers quickly throw code "over the wall" to testers, so that errors are seen as somehow arising during test, rather than from coding.
2. The organization skips the design reviews so that design faults are seen as coding faults.
3. Testers pass code into operations so problems can be classified as "maintenance faults."

These three examples are the type of distribution activity that prevents blame, in response to measurement systems that are used punitively rather than for control activities. When you don't know how to prevent errors, what else can you do but prevent blame for errors? Of course, to the extent that the workers are playing "hot potato" with faults, they have that much less time to do actual productive work. We'll see more about the hot potato phenomenon when we study the dynamics of management pressure.

But not all distribution activities are disguised forms of hot potato. Where blame is not the name of the game, distribution actually serves useful purposes. Pattern 3 organizations tend to distribute the faults earlier in the process than Pattern 2 organizations:

4. Design and requirements work are seen as ways of catching large scope faults early in the development process, rather than later when they will be more costly to resolve.
5. User manuals are written early as a way of revealing faults in interface requirements and generating the basis for acceptance tests. Once again, this unburdens the late parts of the development cycle.

### 1.3. Observational Errors About Errors

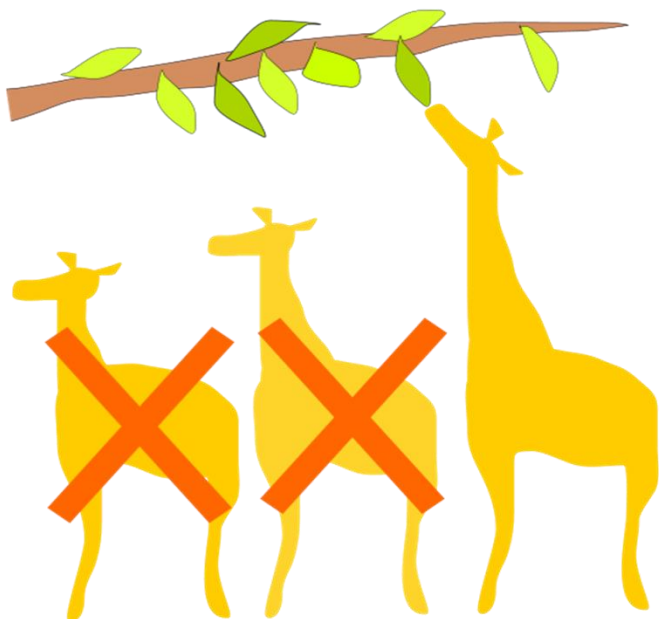
Failure detection is a process of noticing differences—between what is desired and what exists. When we consider the cybernetic model of control, we understand how important seeing differences—failure detection—is to a feedback controller.

Giving things labels is a substitute for noticing. That's another reason I always emphasize the importance of the words controllers use. It's all too easy not to notice important differences if you name two things the same, or to see a difference where none exists if you name them differently.





### 1.3.1 Selection Fallacies



There is a whole class of common mislabelings which I call "selection fallacies."

A selection fallacy occurs when a controller makes an incorrect linear assumption about observation that says, incorrectly, "I don't have to observe the full set of data, because a more easily observed set of data adequately represents it."

It's a fallacy because it doesn't take into account that the processes of selecting the two groups of data may be different, and thus conclusions drawn from one group may not apply to the other. Selection fallacies are easy to spot after the fact, but easy to fall into before, especially if we have some reason to want one conclusion more than another.

#### **Completed vs. terminated projects. Here's an example of a common selection fallacy in software:**

A client surveyed the number of faults produced per thousand lines of code (KLOC) in 152 projects. The study was done very carefully, using the SFA data base for each project. The study concluded that the average project produced 6-23 faults/KLOC, with an average of 14. They felt that this was in line with other organizations in their industry, so they had no strong motivation to invest in further reductions.

Listening to the presentation of this careful study, it would have been easy to miss the selection fallacy, but always being cautious, I asked, "How did you choose the 152 projects?"

"Oh, we were very careful not to bias the study," the presenter said. "We chose every project that was completed in a 3-month period."

"You emphasized the wrong word," I said, now seeing the selection fallacy.

"What do you mean?" he asked.

"You should have said, 'We chose every project that was completed in a 3-month period.' How many projects here are started that never complete?"

The presenter didn't know, and neither did anyone else in the room. I got them to give an approximation, which was later verified by a small study. Historically, in this organization, 27% of initiated projects were never completed. These projects accounted for over 40% of their development budget, because some were not abandoned for a long, long time. A sample of these projects showed a range of 19-145 faults/KLOC, with an average of 38. Later, when the average was weighted by project size, it grew to 86. The two biggest projects also had the highest faults/ KLOC. Where had they gone wrong? In presenting completed projects as representative of all projects, the presenters had committed a common selection fallacy which led the organization to believe that they were not too bad in their fault-producing performance. Then, when they presented all failed projects as typical of their worst projects, they committed the same fallacy in reverse. The second fallacy led them to miss the fact that they simply didn't know how to develop large projects, probably because they couldn't deal with the faults they generated.

## Early vs. late users. Here's another common selection fallacy:

A software organization shipped an update to product X and tracked the STIs that arrived in the first two months. They used these to make a linear projection of the STI load they would have to handle in the following months. Their estimate of the number of STIs was quite accurate, but the total workload generated by those STIs was underestimated by a factor of 3.5.

They had committed several selection fallacies, all based on the assumption that early STIs would be typical of later STIs. They were not because,

1. Later STIs had a far higher failure/fault ratio, because more customers were using the system and encountering the same failures multiple times. The company had no efficient way of resolving these multiple reports of the same failure.
2. Early users of the update were not typical of late users. They tended to be more self-reliant, and worked around a number of failures that later users had to report as STIs in order to get help. Although they were easy to work around, their underlying faults were not necessarily easy to resolve.
3. Early users also tended to use a different set of features than the later users. The typical later user was later because their use was much more extensive, both in features covered and number of people having access to the system. These attributes meant that their installation procedure was more complex, thus slower, which is why they were later users. But more people accessing the system, using more of the features, meant many more STIs.

"He's just like me." The selection fallacy works not only on the observations, but also on the observers. Here's the continuation of the story about Simon, the project manager who couldn't recognize tears.

After Simon asked me whether there was something in Herb's eye, I said, "Well, I really don't know. Why don't you ask him?"

"Oh, it's not really important enough to take the time," Simon replied. "I need to ask you how you think the project is going? I'm really pleased at what a great job Herb did, getting that program ready just a week late, after it was in so much trouble."

"Really?" I said. "I thought you were rather upset about the late delivery."

"Oh, that. Sure, I'd like to have had it on time, but it's no big deal."

"I think Herb thought it was a big deal." "What makes you think that?"

"I believe he was upset when you yelled at him."

"Oh, no. Herb knows me too well to be upset, just because I raised my voice a little. He's just like me, so he knows I'm just an enthusiastic guy."

Simon committed a selection fallacy by assuming "he's just like me." The managers in a software organization are not "just like" the rest of the people—else why were they selected to be managers, and why are they being paid more money? Why not observe the other person instead of assuming the two of you are exactly alike?





Any manager who can't or won't see or hear other peoples' feelings is like a ship's captain trying to navigate at night without radar or sonar. Feelings are the radar and sonar of project life —reflections off the reefs and shoals and shallow bottoms on which your project can run aground. You can't do it with your eyes and ears closed, just using a map inside your own head, if only because you're not just like everyone else.

### **1.3.2. Getting observations backwards**

It's one thing to fail to observe something correctly. It's quite another to observe correctly, but then to interpret the observations backwards, so that black is labeled white and white is labeled black. Some people have a hard time believing that a highly paid software engineering manager could actually label observations backwards, so here a few examples of hundreds I've observed.

#### **Who are the best and the worst programmers?**

A software development manager told me that he had a way to measure who were his best programmers and who were his worst. I was fascinated, so I asked him how he did it. He told me that he sat in his office and observed who was always out asking questions of users and other programmers. I thought this was a terrific measure, and I discussed it with him with great excitement. After a few minutes, however, I realized that he thought the programmers who spend the most time asking questions are his worst ones. I, on the other hand, thought that some of these were probably the best programmers in his shop.

#### **Which is the good quality release?**

When she received her first monthly STI summary for a newly released product, the vice- president of software technology waved it at me and said, "Well, we've finally put out a high-quality release." As it turned out, the release offered so little new function that essentially nobody bothered to install it. Hence, there were virtually no troubles reported. Later, when people did have to install it, they found that it was just as full of errors as all the previous releases.

#### **Why is someone working late?**



A programming team manager told me, "Josh is my best programmer. The reason he starts work in the afternoon and stays late at night is so he won't be disturbed by the less experienced programmers."

It turned out that Josh was so ashamed of the poor quality of his work that he didn't want anyone to see how much trouble he was having.

***to be continued in Next Issue ...***



# 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](mailto:hardpretzel@earthlink.net) or on twitter @JerryWeinberg

**Why Software gets in Trouble** is Jerry's world famous book.

Many books have described How Software Is Built. Indeed, that's the first title in Jerry's **Quality Software Series**. But why do we need an entire book to explain Why Software Gets In Trouble? Why not just say people make mistakes? Why not? Because there are reasons people make mistakes, and make them repeatedly, and fail to discover and correct them. That's what this book is about.

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

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



**TTWT Rating:** ★★★★★



A photograph of a green, conical pendulum bob hanging from a thin wire. The bob is positioned over a surface of light-colored sand. In the sand, there is a large, circular, wavy pattern that resembles a ripple or a footprint. The text "Speaking Tester's Mind" is overlaid on the image in a large, blue, serif font with a white outline.

# Speaking Tester's Mind

- straight from the author's desk



Communicate?  
I am communicating!  
They just don't listen!

***By Ola Hylten***

*Really ? Are you sure about that ? How can you be sure about that ?*

Communication is a difficult art to master. This is well known and has been a topic of discussion since, probably, before the ancient Greek philosophers did their best to argue and debate how to communicate. Communication is a pretty vague word so let me start with trying to explain what forms of communication that I will address here. Primarily I'll talk about communication between team members and team managers but I believe that it is also applicable in other contexts. I will show you how I use some of the communication ideas when testing. If you can make use of it please do.

We talk a lot, some of us more than others, when we work and discuss solutions to testing challenges and problems. In Agile teams it is pretty much mandatory to talk a lot and of course since we communicate largely via the spoken word and body language group forming and group dynamics need the facilitation of communication, and in my experience specifically in the form of dialogue and group discussions. Whether we are consciously aware of it or not this is something that we do in our families as well. We communicate our desires and try to raise children, try to maintain the love between our self and our partner and we read each other's body language and tone of voice very well after a few or maybe a lot of years spent together.

In successful communication within the family, admittedly a fairly well functioning family, we can handle controversy, disagreements and right out fights and, in many cases, bring them to a fruitful conclusion. That's not always instantly or directly after settling something but the fruits of it may be harvested long after, sometimes even years after the original dispute. Of course we all have different ways of handling these situations and they do not always work. We have to adapt our methods and ways of communicating the easy and the difficult messages alike. Does this sound familiar at all? I thought so.

There are no best practices in communication within a family or anywhere else.



We adapt our communication styles according to the different contexts, same as we adapt our testing to the different contexts. So, what have we got here? Context-driven Communication? Well, that's what we do, most of us on this planet I like to believe, and that is part of how we've been successful, in one context, as a species on our planet. It is of course not the only tool but perhaps one of the more important ones through the history of human evolution. We do however have a tendency to forget this in our professional life. OK, not all of us, but I believe that we all are guilty of forgetting to take context into consideration when we communicate at work. If we are to be successful in telling our story and learning I believe communication must be context-driven. If we let context drive the communication then we increase the chance of understanding and learning from each other. To me that's a good thing.



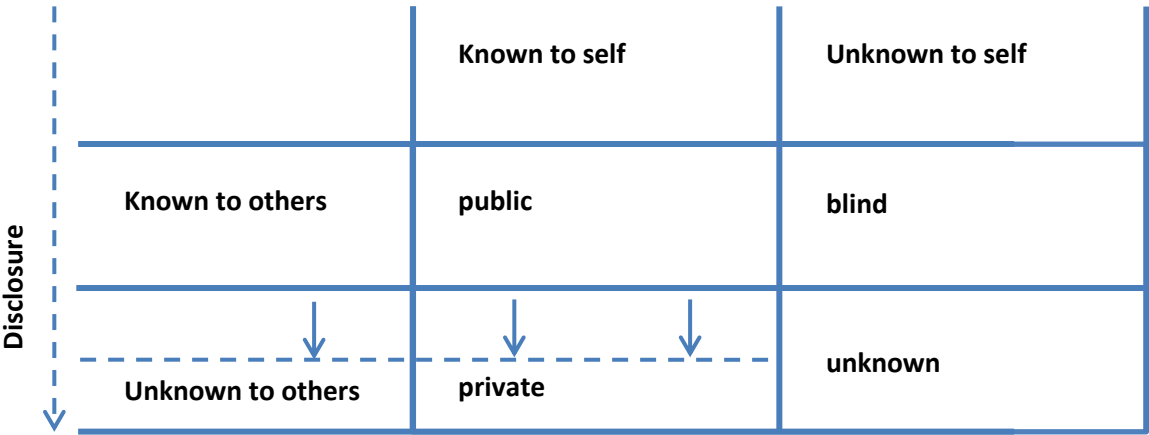
When we communicate there are many differences to consider even in the simple communication between two people talking. What is the professional relationship between the participants? Is one a manager of the other or a team leader for the team? Are they on the same level in the organizational scheme of things but doing different things?

These things affect the communication of course and are obvious if you think about it just a little but the obvious needs to be stated some times to remind us of its impact. We need to remember them in order to evaluate what we hear and construct our responses. We do this without thinking a lot of times. We might not speak as directly as we need to when the other guy is the one who we must negotiate our pay with later. This particular problem is of course very much dependent on the company culture, the country culture and the inter-personal relationship between the persons. In order to minimize the impact of the negative side effects more or less inherent in this relationship it is the manager's responsibility to create a culture and atmosphere of openness. This will help making the, in this relationship, less powerful person feel more secure. One model to illustrate a way to learn more and open up for increasingly better communication between the manager and the persons he/she manages is the Johari-Windown, developed by Luft and Ingham and used by Hersey and Blanchard to illustrate leadership personality and the degree of openness that exists between a manager those under him/her. This is based on the fact that some of a manager's behavior patterns are known only to her/himself, these are private perceptions, some are known to the people managed, to these he/she is blind, and some are known to both parties, these are public.

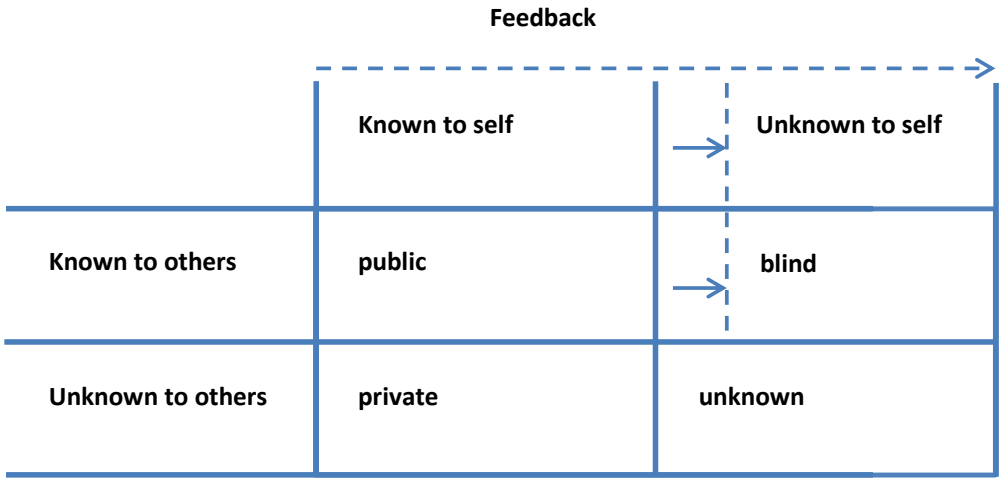
The window therefore looks like this:

	Known to self	Unknown to self
Known to others	public	blind
Unknown to others	private	unknown

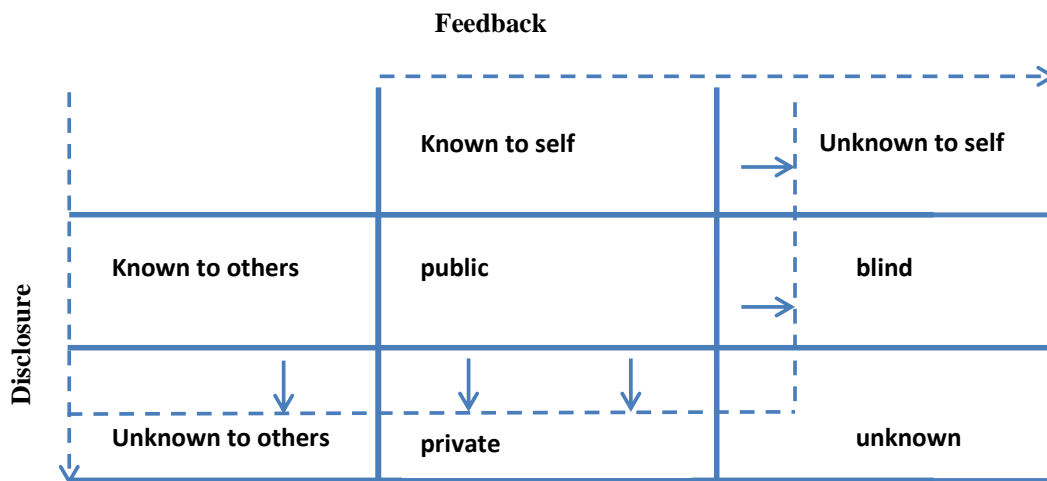
When or maybe if, the manager discloses how he/she feels about something relating to the team and/or business she/he increases the 'public' at the expense of the 'private'.



If the team members give the manager feedback on his/her behavior and how attitude and behavior is perceived by them and, this is not always the case as I'm sure most of you have experience of, is willing to listen, then the 'public' grows at the expense of 'blind'.



So, if we have a manager who is willing to listen and to share her/his own feelings and responses towards the team we will get an aggregated result like this:



We do get into the 'unknown', this alone should be attractive to any tester, because when we communicate openly and allow the 'public' area to grow we will see things that previously was unknown both to manager and team.

How can we get to this state of openness? There are a few things to remember that will make it easier to accomplish this increases of 'public', after all it is in this square we all exist and work together team members and managers alike and we all want room to roam. If I tell you, as a team member to the manager, 'You are so wrong in taking this road to get the new architecture in place', you could, probably quite angrily, reply that I don't know what I'm talking about. This is not helpful in anyway but only increases tension and stress. On the other hand if I say 'I feel concerned about the road map to implement the new architecture' you cannot contradict me since my feelings are something I alone can know anything about and you, if you are willing to listen, will not feel attacked since I have not made a judgment on your decision, I have merely pointed out that I have a bad feeling about it. This will allow us to open up a discussion and increase our chances to achieve our team's goals.

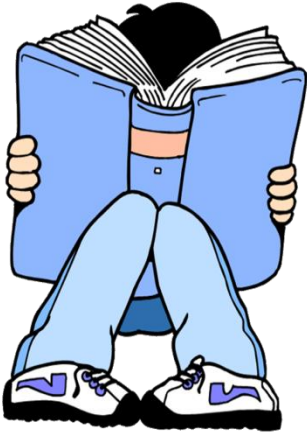
A few points are perhaps needed to clarify the *Johari-window* model and the use and its implications as shown above. First, it relates to business and team related matters. We are talking about a professional situation and what we deal with in our professional life. Second, there may be times when it is appropriate for the manager *not* to disclose anything. This is dependent on context. Third, this is related to managers and groups and there is no 'best' size of the 'public' square in relation to the others. This is dependent on the manager's style and personality as well as on the culture and personality of the team.

One thing that this requires is a manager who is confident in her/his role as manager. Insecurity in the manager role will effectively close the opportunity to expand the 'public' square and will lead to the 'unknown' remaining unknown. Personally I think that would be a shame. I enjoy finding the unknowns in this world.

The idea of communicating like this I also find useful when I've been managing release integrations tests with a collection of different parts or services coming from different projects are being gathered together for a release. Communicating often and early and openly with the projects test managers and architects and I can find unknowns that will help us identifying risk areas so that we can make decisions on where to focus our test effort.

There is plenty more to read and learn from regarding communication within teams and personally I like to dig into my books on sport psychology and on group theory since I enjoy reading them.





Reading about stuff that you enjoy and is curious about is many times a good entry to finding new things you can apply to testing. For me that has been a good road to applying things that I've learned from different fields in my testing. That doesn't mean it will work for you but why not try it!

To me this is a useful model to allow the context I am in to drive the communication.

So how do I put these ideas to use when testing? I'll try to explain my thinking. This is how I reason and try to apply it. When I test I am communicating with the product that I'm testing.

So, entering data, moving things, trying different paths to accomplish different tasks is my way of communicating with the software. In a sense I am actually sharing more of my own thoughts and feelings about the product and so I increase 'public'. OK, the product is not a person, it's a piece of software or something else and it will not respond like a human being, if it did I'd either be very scared or extremely curious to find out how that is possible, so I have to be observant to pick up the feedback that the system gives me. Sometimes that's easy but many times, in my experience, the feedback from the system can be very small and hard to see. Sometimes the first step may be a feeling that something is not quite right here. I can't put my finger on it immediately but I trust my feelings and will try to find out what it is, or was, that initiated that feeling. If I can collect enough 'feedback' then I will find stuff hidden in 'unknown' and that's definitely something I really enjoy doing.

In conclusion I want to point out that this is a model and as all models it's important to remember:

It's only a model! Models are many times useful but it's necessary to point out to people that a model is only a model and therefore incomplete and fallible. It will not be useful in all situations but sometimes people tend to forget that. This model does not take into account language and that is a very large subject and a very important one. It doesn't cover the communication problem of noise interfering with the message on its way from sender to receiver *nor* the issues of the receiver interpreting the message from his/her own context and that can be very different from the senders context thus leading to misunderstanding. Communication is a difficult art to master and I find it important for myself to keep practicing and to keep trying to improve my communication skills. To me it's one of the most fundamental skills I need in order to be able to do my job and to do it well.

Another point is that I have no catchy name for using this model in testing but if you come up with something please let me know.

Thank you for your time.

References :

Team Spirit- the elusive experience, John Syer, The Kingswood Press, 1986

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

[Back To Index](#)



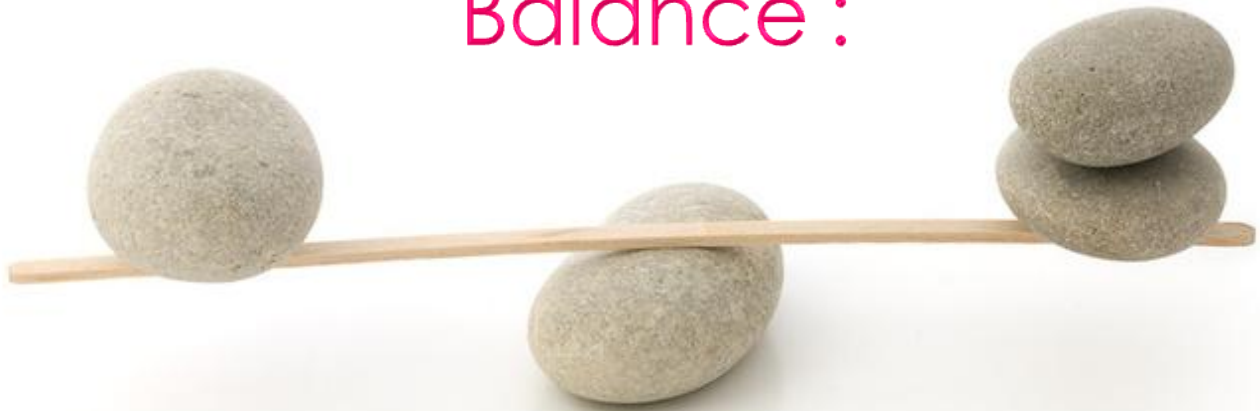
## Biography

**Ola Hyltén** ( Tester, Test Manager, Coach ) :

Ola started working as a programmer in his teens and after an unsuccessful, but fun, attempt at launching a career as a musician he studied Psychology, Philosophy and Computer Science at Lund University, Sweden.

Returning to software development he moved away from programming toward test as his main focus. He likes to think that he thinks for himself and tries to contribute as a part of the Context-driven Testing community.

# Balance :



## Teaching

Vs

## Coaching

***By Olaf Lewitz***

This is about finding and keeping your **balance** as an agile coach or change agent. I introduced the topic as a main theme some time back in *Spotting the Balance*. At the AgileCoachCamp, I committed to writing an article on this. My thanks to *Gitte Klitgaard Hansen* who has been constantly reminding me of that commitment...

I'll start by roughly defining the "opponents".



*A Teacher - by Tim Ellis*

### What is Teaching ?

This is easy. Enabling people to learn. That usually involves some elements of giving them some sort of information (model, tool, technique...) creating a safe environment for experiments guiding them when they make mistakes evaluating their progress

That's enough for the moment, I don't intend to give full definitions here or create a "teaching model"—I'm quite sure you can find that elsewhere. So...

## What is Coaching ?

Coaching is assisting and guiding people to identify the goals they want or need to achieve, to spot obstacles that make it difficult to achieve them, to make out a path to overcome those obstacles and to verify that they stay on that path, so that they actually reach their goal.

That implies a set of Don'ts:

- Don't tell them where to go
- Don't show them the obstacles
- Don't do their work

And as the primary goal is to enable the client to solve their problems their selves, your primary goal is to make yourself dispensable as soon as possible.

This is probably true for a teacher, too...

So where's the conflict?

## Agile Coaching

As agile coaches, we're usually in a situation where we know more about Agile than our clients—that means we have experiences and a toolbox of practices that might help the clients to identify or reach their goal. The goal usually comes down to some variation of "building the right system right"—meaning they want to better identify what system to build, and to achieve better ways to build it.

Which leads to the challenge that while you let the clients define their goal and find their own path to reach it, they'll constantly ask for your advice, your ideas... But you want to stay out of the way.

Notice there's a main similarity and a main difference to a teacher:

- Similar: the topic is set. If you attend a maths class, you don't expect to gain knowledge about biology. When you hire an Agile Coach, you don't expect him to improve your resource management (nor your biology knowledge).
- Different: as a teacher you generally have a predefined curriculum. As an Agile Coach, you have to help your clients identify their learning needs first. At least that's how I understand our job.

So when you're balancing teaching and coaching, you don't want to spoil the one with the other. You need to meet the expectations of your clients to transfer knowledge, as well as your own responsibility to help them find their own way. Notice that failing in one way hurts much less than the other: if you coach while you teach, it probably intensifies the learning. But if you teach when you should be coaching... You might accidentally lead the client to go your way instead of theirs. This is what you want to avoid.



*A Coach - by TheImageGroup*



## Frames

I think to manage this risk it pays to create a clear frame in which to evaluate your options. Your objective is to enable the client (be it the organisation, a manager, a team) to find their own solution. Solution to what? That's your first thing to find out... So we'll take this as an example to establish the concept of your *Coaching Mode* and your *Teaching Mode*.

## Zoom in on the right problem

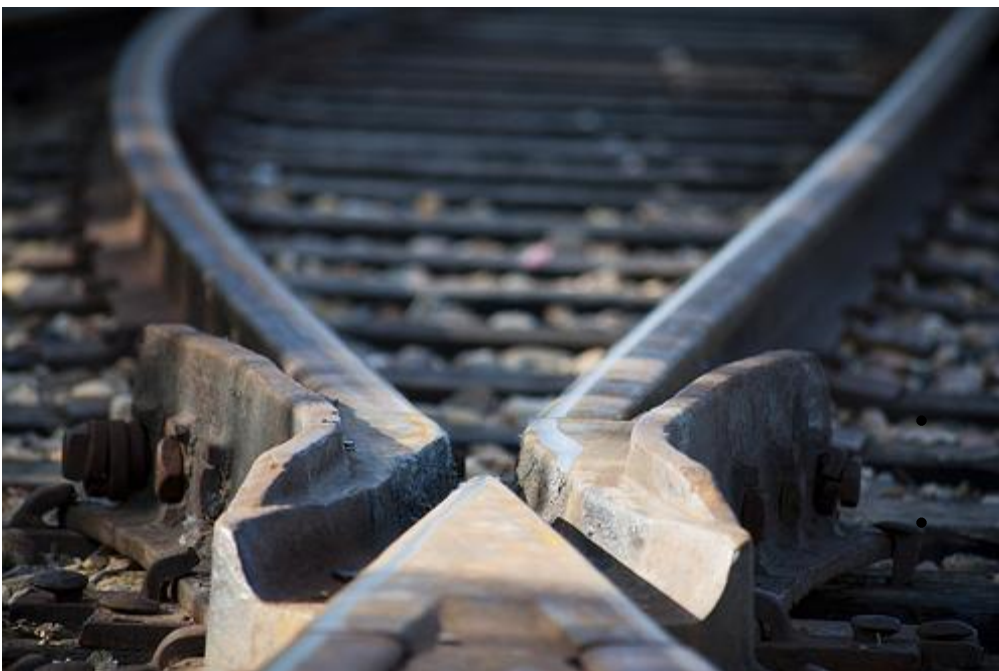
Let's assume your client asks you to teach pair programming. Grabbing the right tool from your agile suitcase, you ask, Why? And depending on the answer, you might ask that question again. They tell you they need knowledge transfer in the teams, as there is a legacy system which only two out of many developers dare to touch, because it's built on an architecture that is very different to the one everybody uses now, and barely recognisable anymore anyway... During this entire conversation you stay in *Coaching Mode*—you ask open questions, listen actively, thereby pulling the tacit knowledge of individuals out into the shared knowledge of the group.

At times, the conversation might get stuck, you get the impression that you're rather talking around the real problem than about it. This is probably your experience, pattern-matching what you see and here with situation you've been in before. Sometimes, just asking another question might not be enough. Rephrasing what you've heard in your own words, trying to break their mental model with an example, can be a good strategy here. When you start giving examples, you enter your *Teaching Mode*.

"I think I've been in a similar situation, when I had to change something in a part of a system where the person who developed it wasn't around anymore, and nobody really knew how it worked... What I felt I needed at the time was..." This lets the group leave their introspection for a while and gives them new information which they can use to reframe their own situation.

You then enter your *Coaching Mode* again and ask, "How is your situation different from my example?" This pattern you then repeat.

## Switching Modes



I think we can generalise for all the balance topics that you need to identify two *Modes of Operation* for yourself. In this case of Teaching vs. Coaching, the *Teaching Mode* and *Coaching Mode*.

This is where your own learning takes place:

- 1.You need to define for yourself how you behave in each mode.
- 2.You need to sense which mode is required in any situation.

3. You need to decide how (and if) you want to make the group know that you're switching modes. Remember that *what they perceive and what you intend them to perceive might not be the same*.

4. You need to hone your switching skills, so that over time it becomes natural to you to adjust your behaviour appropriately to the situation.

## Balancing

In later analyses of other areas of tension where you need to find and keep your balance, I'll expand on these skills. To be successful as a change agent or agile coach, I think you need to master balancing in some of these dimensions, in the sense that you do not need to think about the modes and the switching anymore.

I expect to find other areas of tension which are risky in both directions: Challenging vs. Pleasing for instance is an area where peril lies in both directions (if you apply the *mode* that's not appropriate for the situation). Whereas there will be areas which, similar to this one, can only "fail" in one direction. At first sight, work vs. free time seems to be an area where you do not loose from having too much free time... I'll analyse this further in one of the next posts of the series. Stay tuned.

## Knowledge Injection?

I realised two things while writing this post:

- The usage of the teaching and coaching modes and the honing of your ability to switch between them is not specific to agile coaching, it's certainly applicable to many teaching situations.
- The resulting technique reminded me of BDD and deliberate discovery...

To pay respect to *Chris Matts' Feature Injection* technique we could call this *Knowledge Injection*. Instead of pushing knowledge into the students, you pull the most important knowledge needs from the class, and inject knowledge using examples. This expands the mental model your students have of the problem domain and this model can then again be broken/expanded with more examples (or questions by the students that indicate more knowledge need). This leads to an *Experiential Learning Cycle*. I like that.

## Biography



**Olaf Lewitz** is an Agile Coach at agile42, Berlin, Germany.

As a coach, he inspires people to improve the way they work and the managers to make their organisations more effective.

He has vast experience in Scrum, XP, SPICE, CMMI, V-Model XT, PRINCE2. His main interest is the combination of unusual components and ideas like the implementation of Agile in regulated environments. He has been a software developer, system architect, team leader, project manager, managing director.

Contact him in Twitter @[OlafLewitz](#).



[Click HERE](#)

Discuss this topic on Quality Testing

[Back To Index](#)



# Professionalism in the world of Software Testing

***By Matt Heusser***

In the movie **Man On Fire**, Denzel Washington plays the hero, who takes revenge on the criminals that kidnapped, and claimed to execute, the child he was sworn to protect.

This is not a children's movie.

Instead, it is dark, intense, and a little disturbing. A former US Special Service Operator (it's not really clear what branch of service), Washington's character takes revenge into his own hands, following every lead, and using torture and murder to get to the next person in line.

The bad guys are equally intense -- they know that if they turn in the next bad guy in line, they are certainly dead ... so Washington needs to make it clear they will die right now unless they give the name up.

I'll leave the rest of the story where it lies; both to preserve the surprise and to let people who don't want to hear about it not hear about it.





There is one part of the though, that I found ... fascinating.

As Washington is dealing with thieves, kidnappers, drug dealers, and murderers (most of whom are police sworn to serve and protect), they eventually say the same thing by means of an excuse: "I am a professional."

Now I'm not quite sure what that means, but I think it means something like:

"Hey man, It's not personal. I'm just following orders."

I hope you find the idea expressed above as repulsive as I do.

### **I don't know what professionalism is, but I'm pretty sure that ain't it**

In his foreword to "The Responsible Software Engineer", Tom DeMarco takes aim at this very problem. According to DeMarco, true professionals are required to make, and keep promises -- your doctor protects your medical history, clergy your confession or counseling, and lawyers have attorney/client privilege. In some cases, these promises are implied; the doctor might not explicitly promise to keep your medical records to himself, but he bound to it, nonetheless.

DeMarco points out that this idea of promises can be inverted and turned into compliant uniformity. For an example, DeMarco picks the executive who admits that, yes, the company does drop small amounts of a toxic substance into the water supply every morning ... but to report this to the press would be *unprofessional*.

In this view, unprofessional is code for "doing what I don't want you to do." It's worse, because the executive is abusing the creed of promise-keeping expected of true professionals.

Again, that's not professionalism; is it embarrassment.

### **Professionalism in Our Modern World**

When you look at the history of professional trades (going back to our doctor and lawyer example), they tend to do certain things: To form professional societies, to create licenses, and, in many cases, to create barriers to entry for new people to enter the field. We could argue the merits of these societies at length, but for today, let's just say that when I mention professionalism, that's not what I'm talking about.

In that respect, I'm certainly not alone. My colleague **Fiona Charles** has essentially given up on the term "professional."

In her words:

*"Professional" is a word that has been fuzzed over time, from having a very precise application to wholesale acquisition by every coterie of white-collar workers who wanted more status—or, being charitable—to be taken seriously for their very real skills and (possible) contribution to society. Once, professionals were doctors and lawyers, and maybe engineers. Now, it's apparently you, me, and Harry in the next cubicle who spends his days churning out code. So I don't find "professional" (tester or anything else) a useful label and I can't be bothered wearing it.*

Wow.

Ouch.



Fiona goes on to call herself a craftsperson, which is a fine metaphor.

But I'm not willing to give up on the term professional just yet.

## **Towards A Software Test Professional**

Daedalus, the journal of the American Academy of Arts and Sciences, recently defined a professional as any group for whom the following six statements are true:

- 1. A commitment to the interest of clients in particular, and the welfare of society in general*
- 2. A body of theory of special knowledge*
- 3. A specialized set of professional skills, practices, and performances unique to the profession*
- 4. The developed capacity to render judgments with integrity under conditions of ethical uncertainty*
- 5. An organized approach to learning from experience, both individually and collectively, and this of growing new knowledge from the context of practice.*
- 6. The development of a professional community responsible for the oversight and monitoring and quality in both practice and professional educators*

There is no generally accepted software test body of knowledge; of the few that have been proposed, my school of thought, the context-driven school, generally opposes them vigorously. We do not have a formalized approach to learning from experience, like a doctor's residency program, nor do we have a professional community capable of oversight of its members -- if we did, you might need a license to test, and you could have that license removed. Some of us may have commitments to the interests of our clients and the welfare of society, but that commitment is certainly not universal.

But the capacity of render judgments with integrity under conditions of uncertainty?

We do have plenty of that; we just lack a formal organization.

Maybe we, as the collective group of testers, right now, aren't a profession.

But we, as in you and me as individuals -- can we do better?

I think we can.

## **You**



Right now, today, I can't waive a magic wand and create a test profession. Given the definition above, I'm not sure that I would waive such a wand if I had one.

Yet every day, around the world, some tester is dealing in a situation of ethical uncertainty. He's being asked to skip a test the company promised the customer it would run, or not

file a bug report, or ignore certain behavior, or "just keep your mouth shut" in the meeting with the customer while the big bosses tell the official story.

Perhaps some of those people are reading this, right now.

If you are, let me tell you, are you are not alone. Some of the top people in the field, people like *Ben Simo*, *Selena Delesie*, *James Bach*, and *Pradeep Soundarajan* have faced these sorts of ethical issues -- myself included.

The folks who are on that list faced the ethical issues and did what they believed was right at the expense of a short-term gain, and despite possible short-term pain.

Some faced 0% increases, others were reprimanded ... a few been fired. All of these painful, negative consequences for doing what was right faded over time, to be replaced by something better.

And that is my advice to you today: The one stick we have to push on professionalism, to move our field forward, is to continue to develop our ability to render judgments in situations of ethical uncertainty. (Let's be honest. We know what's right. We gotta *do* it.)

Well, maybe two. Those of us who have that discernment might also band together, be it something like the *Association for Software Testing*, *Weekend Testers*, *Project Sherwood*, the *Miagi-Do School of Software Testing*, or something else.

The business term for this is *differentiation*. In his famous speech at King College, C.S. Lewis called us the "*Sound Craftsmen*."

[Back To Index](#)

Join us.



## Biography

**Matt Heusser** has spent the past dozen years developing, testing, and leading in dev/testing of computer software, lately with an eye toward training and mentoring.

He has been the lead organizer and the first master of ceremonies for the Great Lakes Software Excellence Conference ([www.glsec.org](http://www.glsec.org)) in 2006, and of the Workshop on the Technical Debt Metaphor.

Matt presented a Google Tech Talk on Test Automation in their New York Office during the Google Test Automation Conference.

Matt has shared a column with Chris McMahon in Software Test & Performance Magazine. He currently runs an interview column in Software Test & Quality Assurance Magazine.

His specialties are in Collaboration, Systems and Interactions, Quality, Healthcare/Insurance, Real Estate, Contract Furniture, Web-based software, and Telecom Domains.

Contact him on Twitter @mheusser





Plenty of software testing books tell you how to test well; this one tells you how to do it while decreasing your testing budget.

A series of essays written by some of the leading minds in software testing, **How to Reduce the Cost of Software Testing** provides tips, tactics, and techniques to help readers accelerate the testing process, improve the performance of the test teams, and lower costs.

20% OFF for  
our readers

# How to Reduce the Cost of Software Testing

Edited by Matthew Heusser and Govind Kulkarni

Cam Kanar  
Matt Heusser  
Serena Delisle  
Govind Kulkarni  
Catherine Powell  
Michael Larson  
Michael Burton  
Ed Barkley

Michael Kelly  
Jeroen Rosink  
Karen Johns  
Petteri Lyytinen  
David Gilbert  
Markus Gartner  
Justin Hunter • Gary Beck  
Curtis Stoelkenberg  
Scott Barber  
Matt Heusser  
Catherine Powell  
Jonathan Bach  
Anne-Marie Charrel

Foreword  
What Will This Cost Us?  
The Cost of Quality  
Testing Economics  
Opportunity Cost of Testing  
Trading Money for Time  
An Analysis of Costs in Software Testing  
Test Readiness: Be Ready to Test When the Software Is Ready to Be Tested  
Session-Based Test Management  
Postpone Costs to Next Release  
Cost Reduction through Reusable Test Assets  
You Can't Waste Money on a Defect That Isn't There  
A Nimble Test Plan: Removing the Cost of Overplanning  
Exploiting the Testing Bottleneck  
Design of Experiments-Based Test Case Design  
Reducing Costs by Increasing Test Craftsmanship  
Rightsizing the Cost of Testing  
Immediate Strategies to Reduce Test Cost  
25 Tips to Reduce Testing Cost Today  
Rapid Test Augmentation  
Cost of Starting Up a Test Team

 **CRC Press**  
Taylor & Francis Group  
AN AUBACH BOOK

Enter your Discount Code as

**813DA**

to avail 20% off on CRC Press.

**CLICK HERE** to purchase it right away.

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](mailto:teatimewithtesters@gmail.com) :

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

➤ Your Feedback or Question

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





# In the school of Testing

*for your better learning & sharing experience*



# Learning for Software Testers !



*By Selena Delesie*

The brightest and most successful software testers incorporate self-improvement into their work and life activities. They choose to learn new information and skills on a daily basis. Some choose to contribute their learning back into the testing community.

They **prioritize** time to learn and practice new skills.

## Why Learn?

*"It is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."*

*~ Abraham Maslow*

Intentional learning provides a greater selection of 'tools' to improve quality of work and value of testing. Tools for a software tester can include test approaches, test techniques, domain knowledge, communication patterns, writing techniques, and more. When a tester stops testing for the sake of producing numbers and uses a variety of tools to focus on what is appropriate for a project, they find more bugs, more often, earlier in the project lifecycle.

Self-discovery and learning are critical to my ongoing growth as a tester, leader, coach, trainer and consultant. Continually expanding my toolkit fuels my innovation, increases the value I provide and improves my reputation.

Balancing intended learning between work and personal time improves retention, as does balancing listening and reading with hands-on and practical activities. Both study and practice are needed to maximize understanding and ability in working with a new concept. Regardless of learning preference (visual, auditory, experiential), a variety of approaches helps learn individual concepts and integrate them into existing knowledge and skills. Sharing and discussing concepts further solidifies and improves understanding.

## Time for Study

The accessibility of information means there is no shortage of material to study from. The trick is choosing high quality content that gets you thinking, questioning, and wondering. Some sources will speak to you more than others will, so find ones that work for you.

When selecting subject matter, go beyond software testing and include areas like programming, philosophy, psychology, art, communication, or visual design. Almost any subject will help you expand your testing skills, so the sky is the limit!



## Visual Sources

*"You cannot open a book without learning something." ~ Confucius*

There are many outstanding books on software testing and in other domains that can help software testers learn to do better work. Book stores and online searches are good starting points for finding new books to read. For a broader selection, review your favourite blogs and social media feeds. Personal recommendations and book reviews (particularly by people you respect) often identify the better quality books to read.

Print magazines, online magazines, websites, and blogs are also rich sources of learning. Some provide higher quality content than many books do. These sources allow you to read small chunks of information and focus on an area you want to learn about, or discover a new idea or domain that you never considered. You can easily dig deeper in one area or look broader across interconnected areas. An added bonus is that many are free.

See my [website](#) for some books, websites and blogs I recommend.

## Audio Sources

*"Listening well is as powerful a means of communication and influence as to talk well." ~ John Marshall*

Sometimes listening to someone talk about concepts and domains can engage our brains in new ways. If you have the budget for it, attend a conference, go on training, or take a course. If your budget is

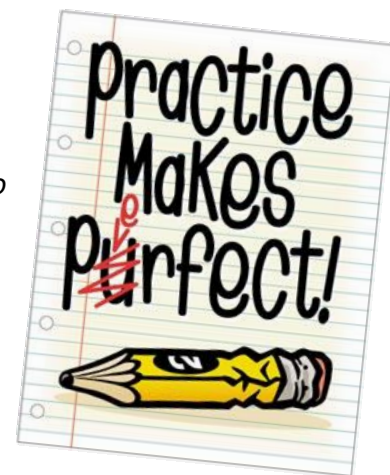
constrained, attend a webinar or conference online. Some are free and most are inexpensive. Either way, you will be able to listen to the speaker and view presentation material concurrently.

Some organizations that provide both in-person and virtual audio learning options include: *Association for Software Testing*, *Software Test Professionals*, *Software Quality Engineering* and *QAI*.

## Practice at Work

*"We learn by example and by direct experience because there are real limits to the adequacy of verbal instruction." ~ Malcolm Gladwell*

That old adage "practice makes perfect" is good advice. While you may not get to perfect, practice allows for integrating new concepts into existing knowledge and skills, experiential learning, and self-improvement.



## Paired Testing



If a colleague is skilled in a test technique, strategy, or domain you are interested in learning, ask them to coach you in a paired testing session. Using one machine while performing an activity with another tester or a programmer forces each person to explain and react to ideas while encouraging creativity. Time-box sessions to stay focused on a specific goal. Paired testing encourages cross-training across technical domains, enhances skill sets of junior team members, supports the evolution of a collaborative environment, and improves team member spirit.

## Testing Workshops

Interactive workshops allow a group of people to focus on practicing different techniques, approaches, and domains. They are easy to set up and can last from 1 hour to a full day. Select a learning goal, choose the software, get participants in a room with a product, and let loose. Pair participants so they collaborate and learn from each other. Spend at least 30 percent of the workshop time debriefing and sharing insights to help participants integrate them into their daily work context.

*One workshop I did was 2 hours long, involved 16 testers, and used two small software applications. Testers paired to practice writing good exploratory testing charters, then did exploratory testing while practicing writing good exploratory session notes. During the debrief at the end, participants worked together to discover what made a charter 'good', what made session notes 'good', and how to improve their exploratory testing approaches.*





## Testing Dojo's

Testing dojo's are similar to workshops, but instead of having everyone paired and working at the same time, one or two people perform activities (techniques, approaches, etc.) while everyone else watches. Every 5-10 minutes, the performer(s) are swapped out for someone else in the group. This approach allows participants to learn from everyone else's ideas and approaches *as they happen*, then improve upon them.

## Practice at Home

Escape the daily work routine to learn and practice new concepts. There are many options for testing products outside your day job that allow you to enhance and refine ideas, skills and knowledge. You can include all of these activities on your CV.

## Weekend Testing

*Weekend Testing* is an opportunity for software testers to collaborate, test various kinds of software, foster hope, gain peer recognition, and be of value to the community. Participants spend a couple of hours on the weekend with a group of people testing towards a specific mission, then discussing their experiences, challenges, bugs, and questions. Chapters have opened up around the world, so look for one near you to get started.

## Alpha or Beta Testing

Volunteer to be an Alpha or Beta tester for a favourite product or in a new industry you want to learn more about. Testing is a complex task, and companies need help getting more test coverage, so there are many opportunities to learn and practice new skills on different software solutions. Check company websites for opportunities or contact the company directly.



## Open Source Product Testing



There is an increasing amount of open source software products and they all need to be tested. Search online to find something that interests you to practice new techniques and approaches, and to work with new types of software and technical domains.

This is another great way to build your reputation and collaborate with people around the world.

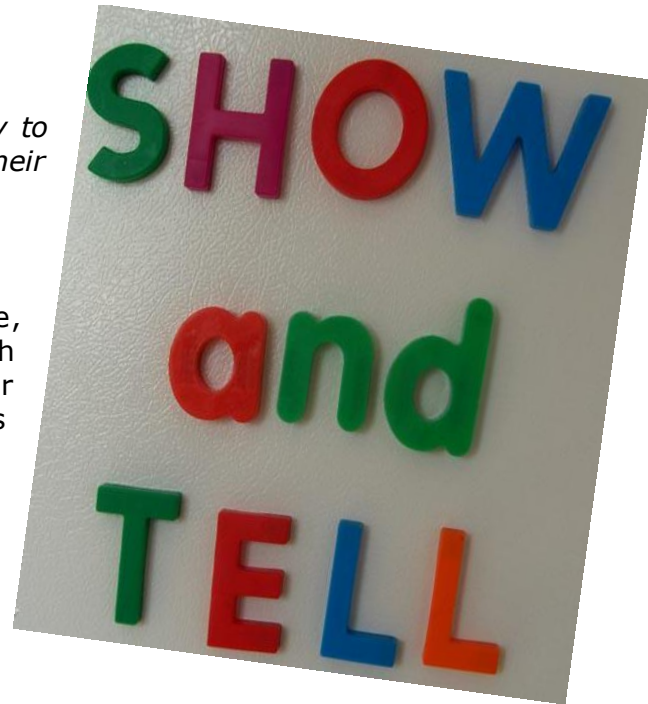
## Testing Communities

Join a testing community for paid testing engagements with real companies. Some may not pay well, but they provide an opportunity to work with different products, try new test techniques, and work with different technical domains. You can also collaborate with testers around the world and improve your industry reputation. Some to check out include: *uTest*, *Software Testing Club's - The Crowd*, and *Mob4Hire*.

## Show, Tell, Discuss

*"Human beings, who are almost unique in having the ability to learn from the experience of others, are also remarkable for their apparent disinclination to do so." ~ Douglas Adams*

Take your learning a notch higher by sharing your knowledge, receiving feedback, and discussing ideas and concepts with others. Explaining your thoughts so others understand your perspective forces you to better understand something, as does receiving feedback and questions on your material and statements. Questioning and debating someone else's ideas will help you think more critically and expand your knowledge base. All can be done verbally or in writing, and online or in-person.



There are many options available:

- Attend a peer group, workshop, or conference in-person or online to discuss, debate and examine a topic in real-time (search online to find one near you).
- Join and participate in an online forum or group to ask questions, or discuss, debate, and examine ideas (search on LinkedIn, Yahoo groups, Google groups, or your favourite testing, software, or technical website).
- Interact with people who are talking about topics that interest you via social media – ask questions, converse and debate (Twitter, Skype, FaceBook, and Google+, to name a few). For example, I have used Twitter to discuss and debate in real-time, learn new things, test in collaboration with testers globally, find new things to try, and meet people globally.
- Write a blog or articles for magazines and websites to share your experiences and ideas. Be prepared to respond to any comments and questions you receive. You can also comment and question on other people's blogs and articles to extend your learning from their writing.
- Share your ideas and experiences by presenting at conferences, peer groups, or webinars. You will be challenged to explain and respond to questions and comments on the spot, which helps deepen your understanding of your domain and topic.

## Take Ownership of Your Learning



Learning new skills and concepts takes time, effort, and practice. So you need to invest time, put forth the effort, and take time to practice. Your learning is your responsibility.

As Eric Hoffer said:

*"In times of change, learners inherit the earth, while the learned find themselves beautifully equipped to deal with a world that no longer exists."*

The benefits far outweigh the costs, as you expand your skill set, improve the quality of your work, reduce time-to-delivery in your tasks, improve your reputation, gain respect from stakeholders, and improve the value testing provides in your organization.

... So what will you learn today?





## Biography

A consulting software tester and agile coach, **Selena Delesie** has been managing and coaching on software, testing, and agile practices for a decade. She facilitates the evolution of good teams and organizations into great ones using individualized and team-based coaching and interactive training experiences. Selena is a contributing author to *How to Reduce the Cost of Software Testing* and an active speaker, participant, and leader in numerous associations and conferences.

Follow Selena online at [DelesieSolutions.com](http://DelesieSolutions.com).



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 !







**David Burns** is a Lead Developer in Test at Mozilla working in the Automation Services team.

He is .Net/Python/JavaScript fan & Selenium Committer.

He has also authored a book on Selenium i.e.

*Selenium 1.0 Testing Tools: Beginner's Guide.*

Contact David on his website [theautomatedtester.co.uk](http://theautomatedtester.co.uk) or follow him on twitter @AutomatedTester .



Learn by doing: less theory, more results

## Selenium 1.0 Testing Tools

Test your web applications with multiple browsers using the Selenium Framework to ensure the quality of web applications.

### Beginner's Guide

David Burns

[PACKT] open source  
PUBLISHING

## Getting started with Selenium WebDriver and Python

I have been the Lead maintainer on the Selenium Python bindings for just under a year now and have seen how a number of people want to get started using the bindings. This tutorial will help you set up a development environment and writing your first bit of Selenium WebDriver code.

To get started you will need to make sure that you have Python installed, if you have Linux or Mac OS X it will already be installed. The next step is to make sure that you have pip installed. This will allow us to install all the libraries that we need with very little effort. To do this you will need to install easy\_install. This is found in the setuptools package.

Next step is to install pip. We do by doing `<sudo> easy_install pip`. The sudo part is needed on Mac OS X and Linux.

The next thing that I recommend that you do is install virtualenv and do it via the virtualenvwrapper package. It means that we can install python packages without polluting the standard packages that the come with Python. This is a standard way to develop with Python.

There is a very good guide available at [doughellmann.com/projects/virtualenvwrapper/](http://doughellmann.com/projects/virtualenvwrapper/).

Next step is to work on your newly created virtualenv.

We can now install Selenium. We do this by doing

- `pip install selenium`

Before we start we need to be aware of one thing, the WebDriver API is idiomatically different to the RC API. The Selenium RC API has all the methods on the Selenium object. This has meant that the API has grown quite organically to 152 methods. The WebDriver API on the other hand is Object Orientated. There is an object that represents the browser, an object that represents the elements on the page, object that represents alerts and so on.

This differentiation of objects keeps the API as clean as possible but allows users to then extend it the way they want. Now that we know this basis we are ready we can easily start writing our first bit of code.

Open a text editor, I use Vim, but Notepad is just as good.

```
if __name__ == '__main__':
```

```
    # let us import the standard libraries we will need from selenium import webdriver
```

```
    # the By class gives us access to the locator strategies we need to find elements on the page
    from selenium.webdriver.common.by import By
```

```
    # we need to create an object to represent the browser, this will start the browser for us.
```

```
    browser = webdriver.Firefox()
```

```
    # Now let us send some commands to the browser
```

```
    browser.get ('http://www.theautomatedtester.co.uk') #note the full url which is unlike Selenium RC API
```

```
    tutorial_link = browser.find_element(By.LINK_TEXT, 'tutorials')
```

```
    tutorial_link.click()
```

```
    browser.quit()
```

If you want to start asserting items on the page there are testing frameworks like unittest or py.test that you can use. At Mozilla the WebQA team use Py.Test as it gives everything that we need. If you want to practice using the python bindings you can always help on a Mozilla project. All of our code is open and freely available.

# Testing...



# with Anurag

## How to test the 3G or Wi-Fi Connection speed on Iphone and Android Smartphones?

### 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](mailto:anurag.khode@hotmail.com)

When you are buying any data plan for your Smartphone, the Tariff and the connection speed are two very basic but most important things that everyone looks for. Many times you pay heavily to your network provider to have good mobile internet connection speed but many times you suspect the speed as even a smaller process takes longer to complete. Well but now, if you wish to check the actual speed your mobile net connection is providing, here is an app for you.



If you are an Android user or you are using Iphone or even Ipad, now you can test the connection speed of your mobile internet connection with very interesting and easy to use app, **Speedtest.net Mobile** by Ookla. Here are some details about this interesting app.

**Application Type:-**Downloadable Mobile Application

**Objective:-**To check the 3G, Edge,Wi-Fi connection speed on your Android and iphone Smartphones.

**Platforms:-** iOS (Iphone,ipad) ,Android

**Network:-**Wi-Fi,3G, Edge

**Price:-** Free

**Size:-** Android-2.1 MB,iPhone-8.0 MB

**Highlights:-**

- This app tests the transfer rates between your smartphone and the closest test server available.
- The easy to use application provides upload and download speed for your network.
- Speedtest.net is the ultimate resource for bandwidth testing and related information.
- The app provides useful information that can be used to compare internet speeds around the country or across different mobile networks.

**How does it work:-**

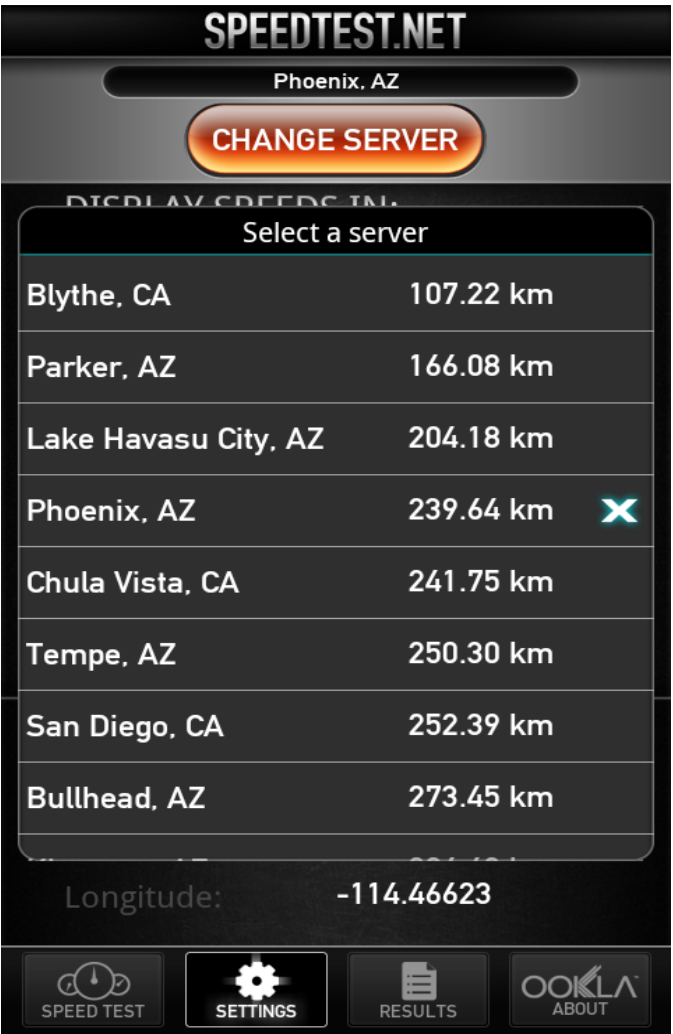
The application is very simple to use.

**Begin Test:-**Press button Begin Test and the application starts working. You can see that the application calculates the download speed followed by the uploading speed with very interactive user interface.

**Different Phases:-**There are three phases: Ping, Download and Upload. The Ping is a test to see how long it takes the test server to respond. The Uploading phase is followed by downloading phase indicating the download and uploading speed of the connection.

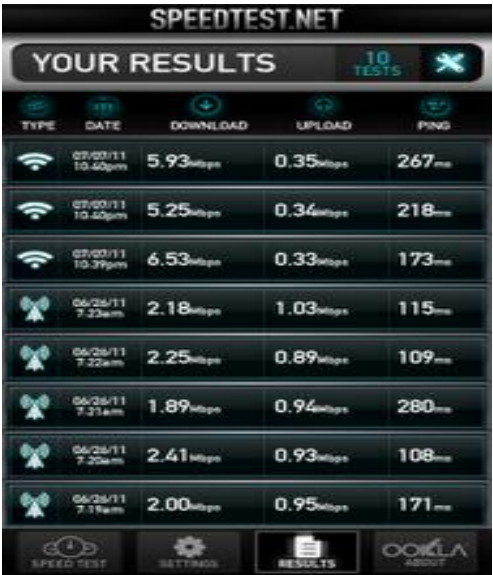


**Settings: Change Server:-** You can change the server from settings tab and select the desired server to verify the connection speed.



**Settings: DISPLAY SPEEDS IN:-** From settings you can choose to display speeds in kbps, Mbps, kB/s

**View Results:-** After several attempts of tests, you can view the complete history of results in the Results tab.



### Tips:-

- Do not rely on only single result. Perform the speed test on different times and take at least 4-5 tests.

### Download Application:-

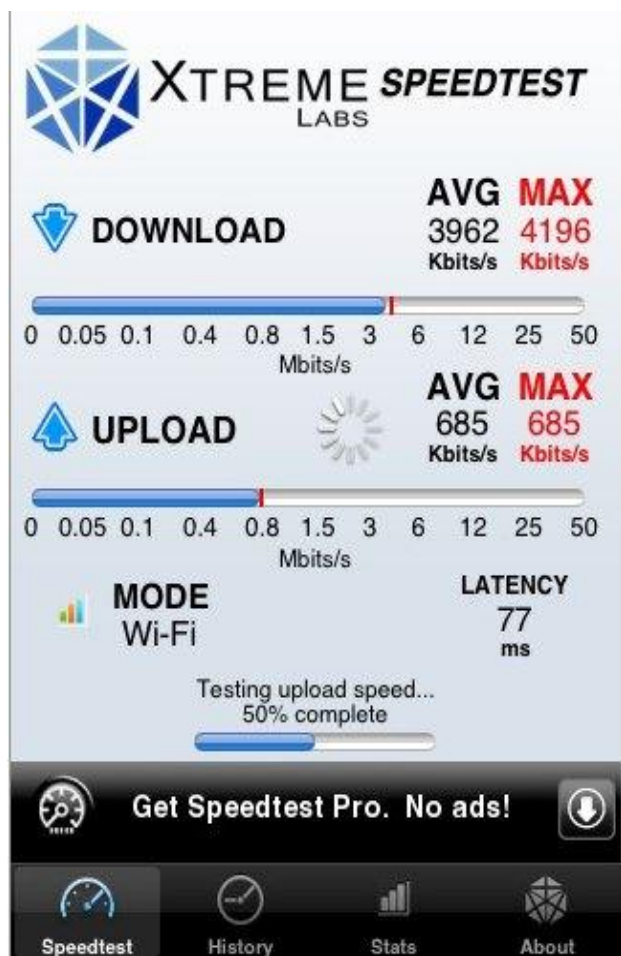
- [Click here to download application from Android Market.\(Android\)](#)
- [Click here to download application from Apple Appstore.\(iphone\)](#)

### Other Similar Applications:-

Speedtest.net provides you one of the most accurate results for the connection speed. However, you still have other similar applications that can work for you.

Here are they:

#### XtremeLabs Speed Test



#### ThinkBroadband(Beta)

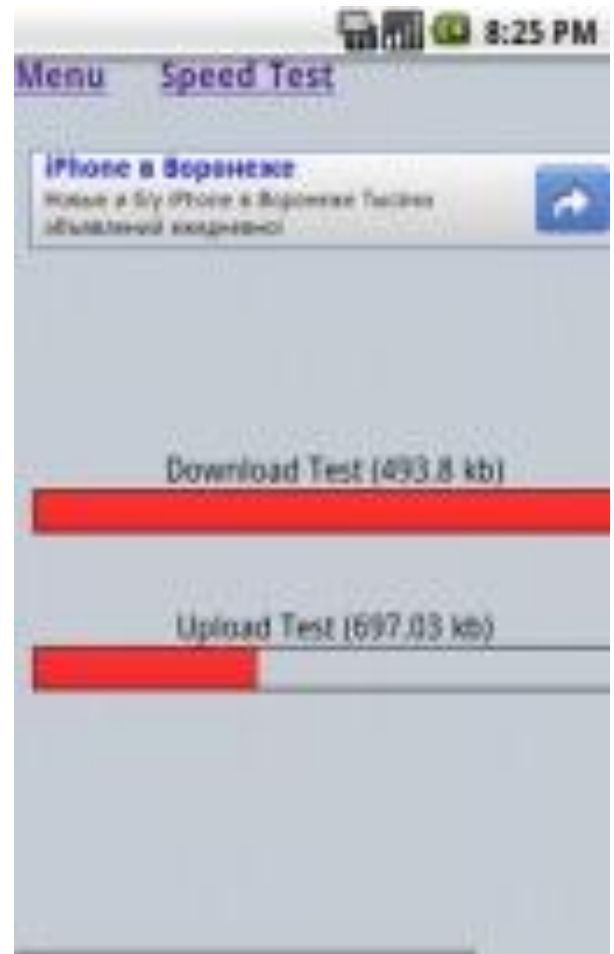




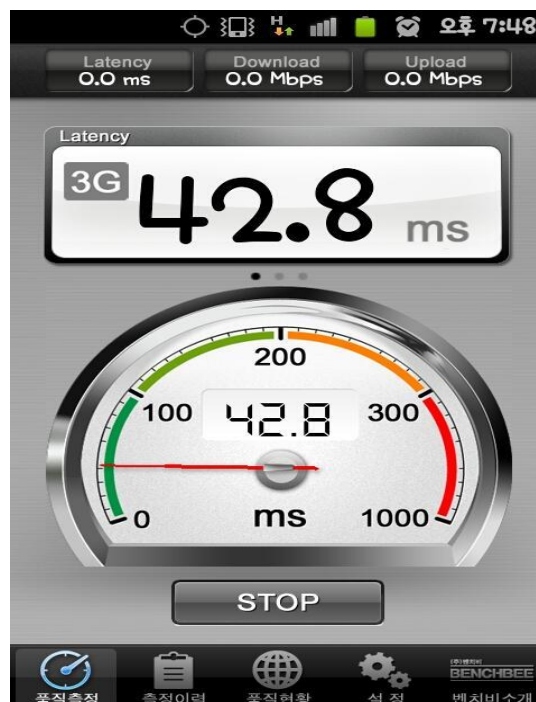
## Inetwork Test



## free SpeedTest BA.net



## BenchBee SpeedTest →



[Back To Index](#)

# testing intelligence

- *its all about becoming an intelligent tester*



an exclusive series by **Joel Montvelisky**

## Manual and automated tests together are challenging

I have a simple question to ask those of you who have any kind of test automation in your teams:

***"Do you have tests that you still run manually, even though they are also running as part of your automation test sets?"***

I am talking about the fact that even though you invested the time to automate something you still choose to run it manually.

Notice that I am asking for *any tests*, including the ones you run manually only once in a while or those you give to your junior testers to be 100% sure all is OK; in fact any tests you are still running both automatically and manually at the same time.



Surprisingly enough, even organizations with a relatively mature automation processes still run a significant number of their automatic scenarios as part of their manual tests on a regular basis, even when this doesn't make any sense (at least on the theoretical level).

After realizing this was the case I sat down with a number of QA Managers (many of them PractiTest users) and asked them about the reason for this seemingly illogical behavior.

They provided a number of interesting reasons and I will go over some of them now:

### **We only run manually the tests that are really important**

The answer that I got the most was that some teams choose tests to be run automatic and manually only when they are "really important or critical".

This may sound logical at first, but on the other hand when you ask what is their criteria for selecting the tests that should be automated most companies say they select cases based on the number of times they will need to run them and also based on the criticality or importance of the business scenario. In plain English, they automated the important test cases.

So if you choose to automate the test cases that are important then why do you still run them manually under the same excuse of them been *really important*...? *Am I the only one confused in here?*

### **We don't trust our automation 100%**

The answer to the question I asked above, of why run the important tests even though they are already automated comes in the form of an even more interesting (and simple) answer: **"We don't really trust our test automation"**

So this basically means they are investing 10 or even 50 man-moths of work and in most cases thousands of dollars on software and hardware in order to automate something and then they don't really trust the results? Where is the logic in this?

OK, so I've worked enough with tools such as QTP and Selenium in order to know that it is not trivial to write good and robust automation, but on the other hand if you are going to invest in automation you might as well do it seriously and write scripts that you can trust. In the end it is a matter of deciding to invest on the platform and be serious in the work you are doing in order to get results you can trust (and I don't mean buy expensive tools, selenium will work fine if you have a good infrastructure and write your scripts professionally).

The alternative is really simple, if you have automatic tests you can't trust because they constantly give you wrong results (either false negative or even worst false positives!) you will eventually stop using them and finally throw all the work and money out the window...



### **We don't know what is covered and what is not covered by the automated tests**

This is also another big reason for why people waste time running manual tests that are already automated, they are simply not aware of which scenarios are included on their automation suite and which aren't. In this situation they decide, based on their best judgment, to assume that "nothing is automated" and so run their manual test cases as if there was no automation.

If this is the case, then why do these companies have automation teams in the first place?

### **The automated tests are the responsibility of another team**

Now for the interesting question, how come a test team "doesn't know" which scenarios are automated and which aren't? The most common answer is that the tests are been written by a completely different team, a team of automation engineers, that is completely separate from the one running the manual tests.

Having 2 test teams, one manual and one automatic, is not something bad and in many cases it will be the best approach to achieve effective and trustworthy automation. The bad thing is that these teams can sometimes be completely disconnected and so work on the same project without communicating and cooperating as it should be.



I will talk about how to communicate and cooperate in a future post, but the point here is that when you have 2 teams (one automated and one manual) you need to make an extra effort to make sure both teams are coordinated and as a minimum each of them know what the other is doing in order to plan accordingly.

### **We want to have all the results in a single place to give good reports**

Finally, I wanted to mention a reason that was brought up by a number of test managers, even though they brought it as a difficulty and not a show stopper but it was brought up many times and so it sounded interesting enough to mention it. The fact that they needed to provide on unified testing report for their project, and for this they either run part of their tests manually, or created manual tests to reflect the results of their automation.

Again, this looks like a simple and "relatively cheap" way of coordinating the process and even producing a unified report, but it suffers from the problem of being a repetitive manual job that needs to be done even after you already have an automation infrastructure and it will eventually but surely (specially as more and more automation is added) run into issues of coordination and maintenance that will make more expensive and in some cases will render it misleading or even obsolete.

## What's your take?



I am actively looking for more issues, experiences or comments like the ones above that revolve around the challenges in manual and automated testing. Do you have stuff you want to share? Please mail me directly to [joel@practitest.com](mailto:joel@practitest.com)

We've been working on a solution for these types of issues and so we are looking for all the inputs we can get in order to make sure it will provide an answer to as many of the existing challenges as possible. I will be grateful for any help you can provide!

## 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](#)

## Teach-Testing →

An Ambitious Campaign by Tea-time with Testers



Click here to Cast Your Vote

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




by



Back To Index





# 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](mailto:teatimewithtesters@gmail.com).

Submit your articles, stories, thoughts around software testing.

## now its your chance to be heard...!

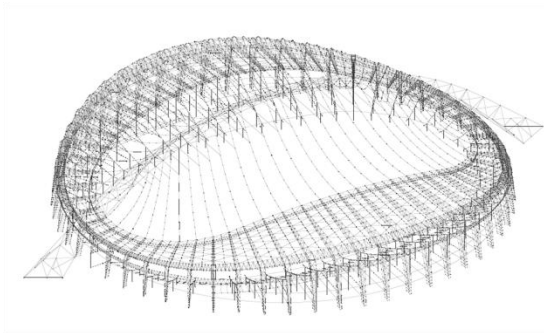


# T ' Talks



*T. Ashok exclusively on software testing*

## **The mistaken notion of Structural Testing**



Whenever I conduct a session on testing, I am always amazed at how misunderstood structural testing is. Whenever the topic of white box comes up, the participants, typically full time test professionals involved in testing the whole system, say that this is about testing the code by executing the various statements, branches or paths. The immediate conclusion is that this is to be only done by the developer and therefore does come under purview. When probed further they say "white box testing" is about doing coverage based testing.

Hmmm.. Let us think on this..

Let us commence from discussion on the phrase "white box testing"- Sadly the phrase as used is incorrect, it should be "white box test techniques". That is, a set of techniques that use the structural properties of the system to evaluate behaviors. The premise is - the system could fail because of faults in the structural aspects of the system.

So what are the structural aspects that we would like to examine? Before we commence on this line of reasoning, let us ask a basic question as to the meaning of structure. What are the elements that constitute structure? What are properties of the structure that we want to evaluate?

The set of parts and their linkages constitutes the structure. A part in turn is built using some “building materials”. Our intention is to ensure that the structural aspects of these parts and their linkages are correct, providing us with a robust scaffolding to deliver functional/non-functional behaviors.

### So what a “part” mean in our context?

The building materials would be the technology/languages at lowest level, components/subsystems at the middle level and other systems/environment at the highest level. Therefore what a part is level dependent, a part could be API/object at the lowest level, whilst it could be subsystem/tasks/services at middle level and other systems with we have to talk to at the highest level.

### What about “linkages”?



Only when all the parts are “connected”, do we get the whole system. The linkage is done in two ways - calling & sending i.e. calling a API/service and sending data. Think about this - when you need help from a friend, you can call on the phone or send a text message. So the linkage could be a function/method call, sending data via messages or sharing the data via global/environment variables/files/databases.

So let us connect all these. We want to ensure that the materials that make up the parts have been used correctly and that the parts have been linked correctly. At the lowest level, the part is constructed using flows that are sequential/recursive/parallel and using materials from the environment i.e. resources. These parts are then linked with parts by interfaces that are call/data based. But it is important to keep in mind as to what “level” the part is - lowest level code structure, system architecture at the middle level and deployment architecture at the highest level.

Employing this line reasoning, do see that as test professionals it is our bounden duty to ensure that the structure is indeed robust and that we have to validate correctness of the structure at three levels:

1. Understanding the deployment architecture and looking for issues in at the higher order structure. Do we know dependencies on other systems, data formats of linkages, latencies, environment resource availability, the plethora of versions of the environment etc.
2. Understanding the system architecture in terms of flows of control/data, sequences of usage, error possibilities and exception handling keeping in mind the non-functional aspects like performance, security attributes too.
3. At the lowest level, understanding the code structure in terms of control and data flows and resource usage. This may be difficult to do for a full time test professional as this may require a higher level of detail of code structure and hence may be better done by a developer. Nonetheless what needs to be validated can be hypothesized by the tester and the act of execution delegated to the developer.

Let us discuss two scenarios to illustrate this..

**Scenario #1 :** Joe goes to the doctor after a rough night and states that he was feeling very uneasy, with a mild pain the chest, had excess sweat and felt pretty tired. The doctor after external examination comes to a conclusion that he needs to be admitted for a by-pass surgery. No way!

**Scenario #2 :** Joe goes to the doctor after a rough night and states that he is unable to breath normally when sleeping. He says that he breathes though mouth and gets up in the morning with a sore throat. The doctor says that he needs to a detailed examination by cutting open the nose in the middle to uncover the problem. Ouch!

In scenario #1, the act of probing is done via external examination whereas in scenario #2, the examination is intended to be internal! I am sure you would wince at both these scenarios. You would probably expect a meaningful and judicious combination of external and internal examination.

That is what is expected from us too! So never think in terms of pure *black or white*, it is always *black and white*. Look at the structure from three levels and when you require a detailed examination of the lower level structures, work with the developer. And for heaven's sake do not use the term "White box testing".



Remember the structure is very important as it provides the scaffolding for the system and needs to be evaluated. A poor structure has a significant bearing on the correctness of attributes like performance, load handling, scalability, reliability, security etc. Note functional correctness may be still be delivered with a poor structure, it is a just a matter of time before cracks show up!

I am sure you will agree with me that the probability of faults in a system is directly proportional to the complexity of the system. Some systems are functionally complex (i.e. number of conditions) whilst some systems are structurally complex and some systems are attribute-wise complex (i.e. need to deliver a simple functionality but may need to support of millions of instances). Understand that structure plays a vital role in the last two.

In **HBT (Hypothesis Based Testing)** the three core concepts of Quality Growth Principle and Complexity Analysis and Techniques Landscape enable one to scientifically understand the complexity, when and how to evaluate the structure.

Remember, you are also responsible for structural aspects of the system. If the building falls, you will be famous and you probably don't want be there on that pedestal!

Cheers!

[Back To Index](#)

**Note:** Drop me a note at [ash@stagsoftware.com](mailto:ash@stagsoftware.com) or tweet me [@ash\\_thiru](https://twitter.com/ash_thiru) if you liked this. Thank you.



## 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](mailto:ash@stagsoftware.com) .







## OUR PARTNERS

## 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.



A black and white photograph of a man in a suit and glasses looking through binoculars. The image is framed by a thick black border. The text 'Tool Watch' is overlaid in a large, white, sans-serif font.

# Tool Watch

about various testing tool around

# StressTester<sup>TM</sup>

## PART 3

Tea-time with Testers Rating: ★★★★★

by Juhi Verma &  
Sharmistha Priyadarshini

### 1) Archive Your User Journey



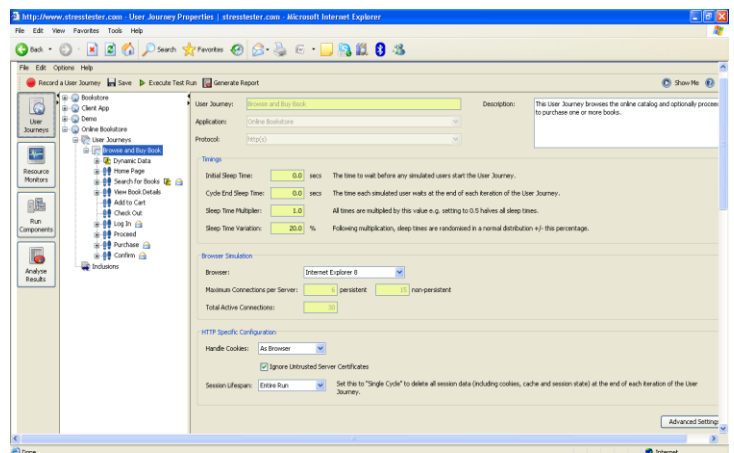
Once your User Journey is running as expected, it is advisable to make a back-up of it so that if any further configuration introduces problems, you can always revert to a working copy. To make a back-up within the GUI, right-click on the User Journey node and select the 'Archive' option. StressTester will make a new archive copy of the User Journey. To see old archives, within the Options menu in StressTester, select 'Show Archived User Journeys'. To hide old archives, unselect this option.

If you wish to revert to an old archive, right-click its node and select 'Make Active Version'. This will archive the current version of the User Journey and then copy the selected archive to be the new current version.

### 2) User Journey Properties

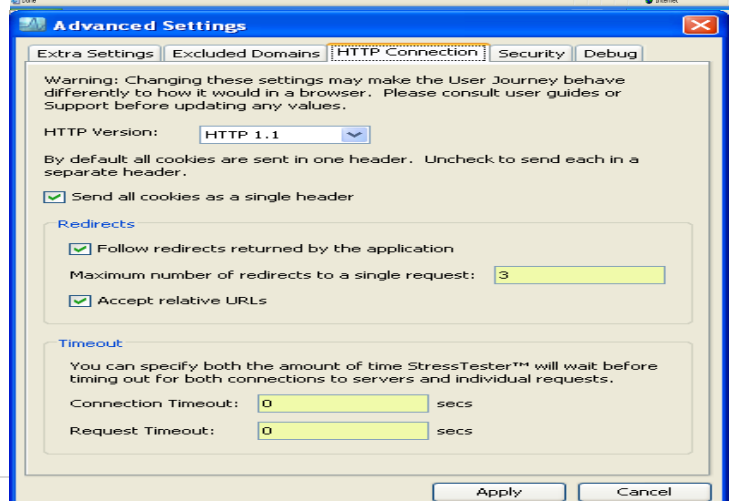


When you select a User Journey node in the navigation panel, the User Journey's properties will be displayed. Clicking 'Help' will show detailed help on every field on this screen. The key fields to consider on this screen are those related to the browser to simulate, and how the HTTP protocol should work when the User Journey is executed.



### User Journey Advanced Properties

By clicking the Advanced Settings button on the User Journey properties screen, you can access other properties that rarely need to be changed. You can obtain information on the purpose of all advanced properties by clicking the 'Help' button on the User Journey properties screen.
















### 3) Step Icons

Following a Step Group's or step's name in the navigation panel you may see one or more 'quick glance' icons. These icons indicate configuration on the Step Group / step. The table below describes each icon.

	Indicates that Success Validation has been specified for this step which means that the content of the application's response will be checked to ensure it contains the expected data.
	Indicates that the request will be an HTTP POST request. Such requests normally correspond to web pages where data has been entered by the user into fields and forms.
	Indicates that the step uses one or more Dynamic Data items (parameters).
	Indicates that the step has Flow Control configured (meaning that the default flow through the User Journey may change after this step has executed successfully).
	Indicates that the Step Group is configured to use the Dynamic Content feature. Once the lead step has been requested, any content referenced within the application response will be automatically requested by StressTester (in the same way a browser loads a web page).
	Indicates that the response of the step will be passed to external Java code for further processing.
	Indicates that a scriptlet has been configured to run before and/or after the step is executed.
	Indicates that the step will only be executed for the first cycle of the User Journey for each simulated user.
	Indicates that the step will only be executed for the last cycle of the User Journey for each simulated user.
<b>Tx</b>	Indicates that this step, if it executes correctly, will be counted as a transaction when StressTester calculates transaction throughput information.



### 4) Search Property

StressTester will use your criteria to search for matching strings in the following step properties:

- URL
- URL Parameters
- POST data
- POST Parameters
- Success Validation

If you choose the 'Find All' option, every step that matches the search criteria will be selected in the navigation panel, allowing you to edit in multi-step mode or perform some other action on all the matching steps.

## 5) Response Validation



It is very common, when a system is under heavy load, that it will continue to respond to requests but the responses will be incorrect or incomplete.

StressTester automatically performs the following checks on every response:

- No network error has occurred
- No protocol error has occurred (e.g. no HTTP error code)
- The response is complete and was not truncated

If all above are successful, if specified, the value of the step's Success Validation field will be used to validate that the response contains the expected results. Responses can be validated in two ways: against a plain text expression or with a Regular Expression.



## 6) Sleep Times

Every type of Flow Control allows you to specify a time delay that will occur before the destination step is executed. It should be noted that the Sleep Time on the Flow Control configuration is in addition to any Sleep Time specified on the destination step. The Timing fields on the User Journey properties screen affect how the step Sleep Times are varied during the performance test.



## 7) Dynamic Data

Dynamic Data is the StressTester feature that allows you to vary the data sent to your application in the same way as it will vary in the real world. A Dynamic Data item can be thought of as a parameter. Dynamic Data can be referenced in the step property fields listed below:

- URL
- URL Parameter values
- POST data
- POST Parameter values
- Success Validation
- Flow Control
- HTTP Header values (Advanced Settings)
- Authentication fields (Advanced Settings)

## Inserting a Dynamic Data Item

To use a Dynamic Data item you just replace the recorded text with a reference to the item.

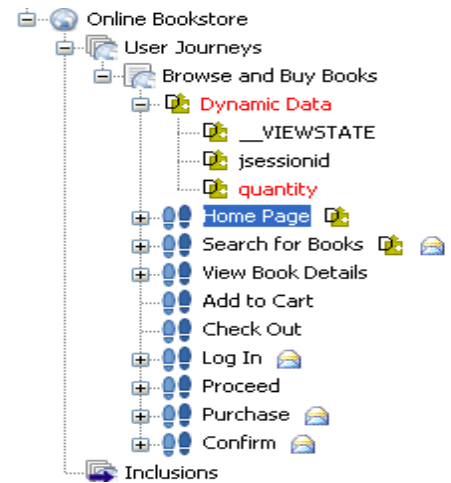
For example, to change the following URL that is searching for books that match "performance testing":

```
www.acmebooks.com?searchTerm=performance%20%testing
```

so that during a performance test the search term will be varied, you would replace the recorded literal with a Dynamic Data item reference:

```
www.acmebooks.com?searchTerm={{searchTerm}}
```

You can either directly type the Dynamic Data item's name where it is needed, or you can use the Dynamic Data wizard to either insert references to existing items or create and reference new items.



When you type the name of the item where you need it, if the Dynamic Data item is not yet defined within the User Journey, the Dynamic Data top-level node and the node of the item that is yet to be defined will be displayed in red font in the navigation panel.

If this happens, you should select the Dynamic Data item's node and configure the item.

The easiest way to replace recorded values with Dynamic Data items (including the configuration of new items) is to use the Dynamic Data wizard.

## Types Of Dynamic Data Type:

- **Auto-Increment**

The Auto-Increment Dynamic Data type allows you to create unique values that include a changing number (incrementing or decrementing by a set amount). You can also configure the generated values to be padded to a certain length, as well as specifying optional prefix and suffix strings.

This type is ideal when creating unique values that contain a number; for example, it can be used to create email addresses of the following format:

```
user0001@acme.com  
user0002@acme.com
```

The wizard screen above allows you to configure your Auto-Increment item.

The Starting Value and Increment must be integers, although the Increment can be a negative value (and hence cause the run time values to decrease).

The Minimum Length field refers to the generated number; if the generated number is shorter than the Minimum Length value, the number will be prefixed with the relevant number of leading zeroes.

The Prefix and Suffix strings are applied to the number, after padding, to form the value of the Dynamic Data item.

When you have configured the item to your requirements, click 'Finish' to save the item.



## • Date

This type of Dynamic Data is of use when the application expects a date to be provided.

The Date Dynamic Data type allows you to generate a date, relative to another date (including another Date Dynamic Data item), in the format your application requires.

### StartingPoint

A Date item can be generated relevant to:

- 'Now' - the date and time the item is used in a performance test (to millisecond precision). Every time the item is used in a performance test the starting point will be re-calculated.
- 'Today' - the date the item is used (to day precision)
- 'Fixed Value' - a fixed date specified by yourself, or
- 'Another Date' - this item will be generated based on the value of another Date Dynamic Data item

If you specify a fixed value its format must match the value of the Format field.

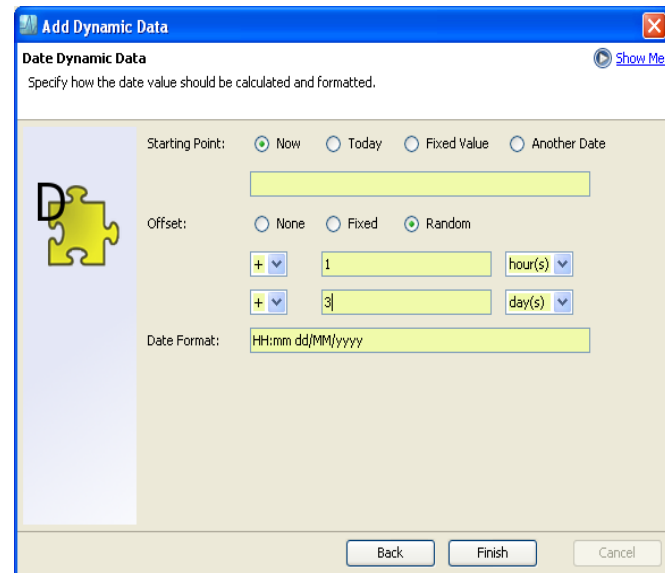
If you choose to base this item on another Date item, a drop-down list will appear showing the valid Date items (an item can only be related to another Date item that itself is not related to any other Date item).

### Offset

Once you have specified the 'starting' date, you can then decide how the Dynamic Data item's value should be generated. You can specify a fixed offset (i.e. the generated date will always be a fixed amount of time before or after the starting date), or a random offset (the generated date will be between two values that are based on the starting date).

### Format

The last thing to define is the format of the generated date to use within the User Journey.



**Add Dynamic Data**

**Date Dynamic Data**  
Specify how the date value should be calculated and formatted.

Starting Point: ☒ Now ☐ Today ☐ Fixed Value ☐ Another Date

Offset: ☐ None ☐ Fixed ☒ Random

+ 1 hour(s)

+ 3 day(s)

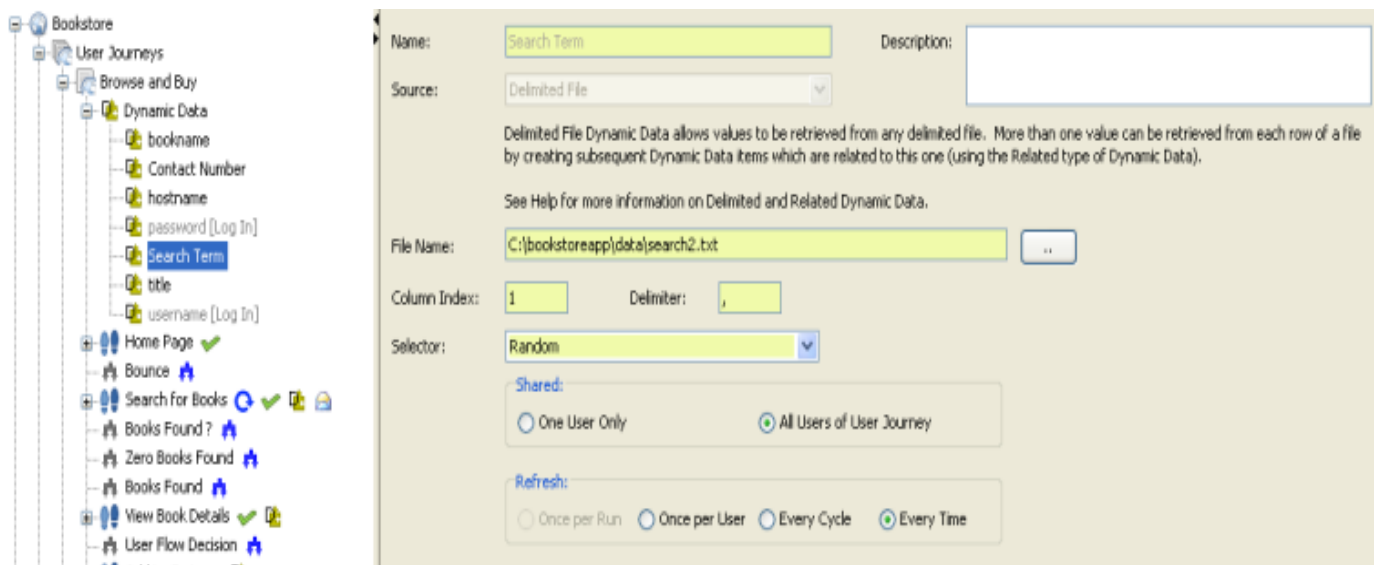
Date Format: HH:mm dd/MM/yyyy

Back Finish Cancel

## Inspecting Dynamic Data Items



You can view and update an existing Dynamic Data item's configuration by selecting its name under the User Journey's Dynamic Data node in the navigation panel.



**Bookstore**

- User Journeys
  - Browse and Buy
    - Dynamic Data
      - bookname
      - Contact Number
      - hostname
      - password [Log In]
      - Search Term**
      - title
      - username [Log In]
    - Home Page ✓
    - Bounce
    - Search for Books ✓
    - Books Found ?
    - Zero Books Found
    - Books Found
    - View Book Details ✓
    - User Flow Decision

**Name:** Search Term **Description:**

**Source:** Delimited File

Delimited File Dynamic Data allows values to be retrieved from any delimited file. More than one value can be retrieved from each row of a file by creating subsequent Dynamic Data items which are related to this one (using the Related type of Dynamic Data).

See Help for more information on Delimited and Related Dynamic Data.

**File Name:** C:\bookstoreapp\data\search2.txt

**Column Index:** 1 **Delimiter:** ,

**Selector:** Random

**Shared:** ☐ One User Only ☒ All Users of User Journey

**Refresh:** ☐ Once per Run ☐ Once per User ☐ Every Cycle ☒ Every Time

Screens similar to the ones used in the Dynamic Data wizard will be displayed and you will be able to update certain aspects of the Dynamic Data configuration.



## 8) Dynamic Content

When a browser downloads a web page's HTML, it will parse the page for references to included content (such as images, cascading style sheets, etc.) and will then automatically download these items.

Dynamic Content is the StressTester feature that allows you to specify, for one or more Step Groups in a User Journey, that StressTester should behave in the same way a browser does: parsing the response to the lead step in the Step Group and subsequently download any embedded content.

So, why is this not the default action for every Step Group in a User Journey?

The answer is to do with the extra processing that StressTester would need to execute for every web page response, for every simulated user.

StressTester has been designed to be very scalable (clients have used the tool to simulate over 100,000 concurrent users) whilst at the same time not requiring large amounts of hardware to simulate high numbers of users.

Therefore, to keep the hardware requirements for machines hosting StressTester Injector processes as low as possible, StressTester by default does not parse application responses to determine which content should be downloaded next. By default it requests the steps configured within the Step Group.

### Note:

Other dynamic data types are: Constant, Delimited File, JAVA Class, List, Related, Response, Variable etc.



## 9) Flow Control

Flow Control is the StressTester feature that allows you to simulate users taking different routes through a business transaction.

For example, for an online ecommerce site, some users will simply browse but not buy anything; others may search for a specific item, add it to their shopping cart and immediately buy it; whereas other users may look at many items and add a random number to their shopping cart.

All of the above scenarios are the same business transaction: 'Browse and Buy'.

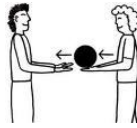
With Flow Control it is easy to configure many different routes through a single User Journey and therefore simulate real world usage of an application as closely as possible.

### When Should Flow Control Be Used?

There are endless situations when Flow Control can be used, including:

- Simulating a percentage of users leaving a web site at different points in the User Journey e.g. it is typical that somewhere between 20% and 60% of users leave a web site after viewing the home page.
- Randomising the number of times a loop occurs e.g. the number of products a shopper views
- Controlling the actions the simulated user takes depending on the results of a previous step e.g. if no products are found, do not proceed to attempt to look at a product's details but instead perform another search with different criteria

## 10) Importing/Exporting User Journeys

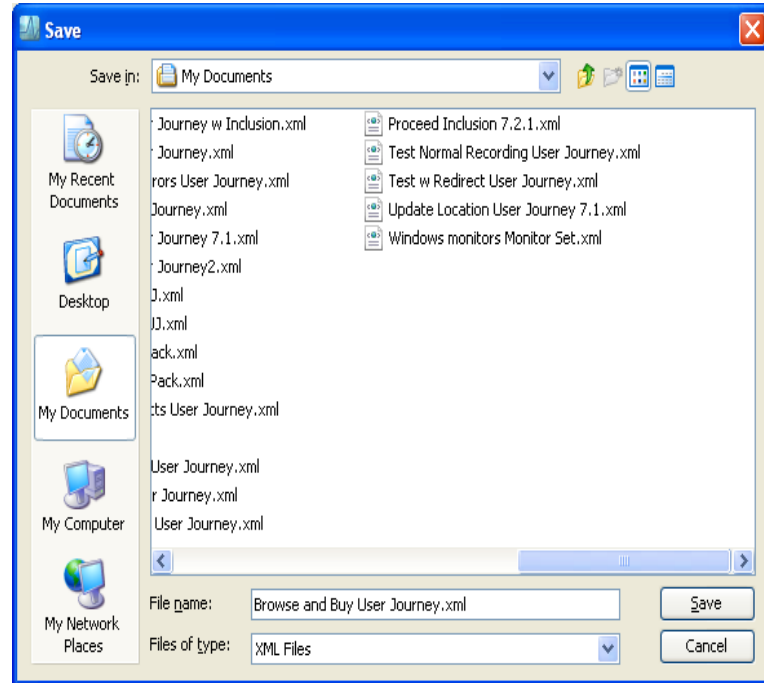


You can move User Journeys between StressTester installations by exporting the User Journey from one installation, and then importing it into another installation.

To export a User Journey, right-click on the User Journey node in the navigation panel and choose 'Export to File'. StressTester will then prompt you for a location and name of the file to create. The export format of the file is XML; although this is easy to read, it is not recommended that export files are modified in any manner unless under the instructions of Support.

An advantage of the file being in XML format is that you can save it in your source code repository thereby storing the User Journeys that test a version of an application alongside that version's source code.

To import a User Journey into a StressTester installation, on the File menu choose 'Import from File' and select the XML file containing the User Journey you wish to import. StressTester will validate the file, import it into the installation and select it within the navigation panel.



Hi Juhi and Sharmistha,

I got your mail id from your article posted in Tea-time with Testers.

It's a good article posted by you guys on "Stress Tester" .I have read part 1 and part 2 as well.

Just a small simple question, is it an open source tool or a licensed one?

I want to explore it for one of our application.

Waiting for your feedback.

Thanks,

Rupali.

Dear Rupali,

Thank you for your kind words. Due to space constraints we have answered your query on our blog site. You can check it out just by clicking [here](#).

Hope it helps. Feel free to write us back if you need more information.

Thanks,

Juhi and Sharmistha





## 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](mailto: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 [sharmistha105@gmail.com](mailto:sharmistha105@gmail.com)

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

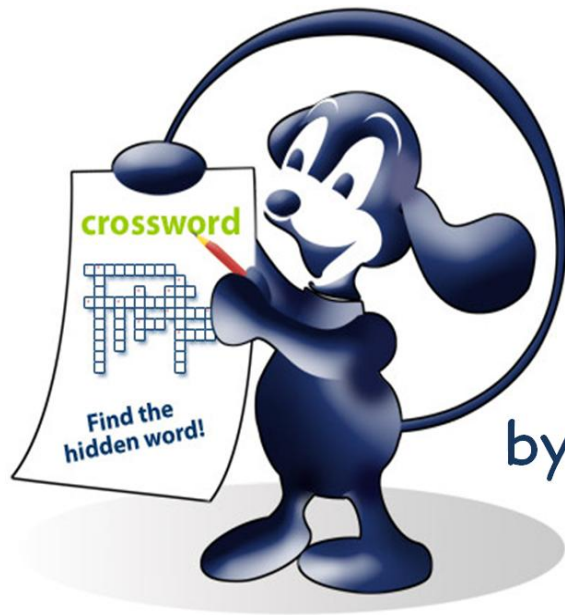
Feel free to write us. Let the world know what your Tool can do !

To know more write to us at [teatimewithtesters@gmail.com](mailto: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 b4 10<sup>th</sup> Oct 2011 & grab your Title.

Send -> [teatimewithtesters@gmail.com](mailto:teatimewithtesters@gmail.com) with Subject: Testing Puzzle

Gear up guys.....

**It's Time To Tease your Testing Bone**

# Puzzle “Domain Name”

"Find the domain with the longest string that you can find. Example of valid domain: "test.com". Subdomains are considered invalid.

The domain should not be redirecting and should have a valid web page."



## 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.

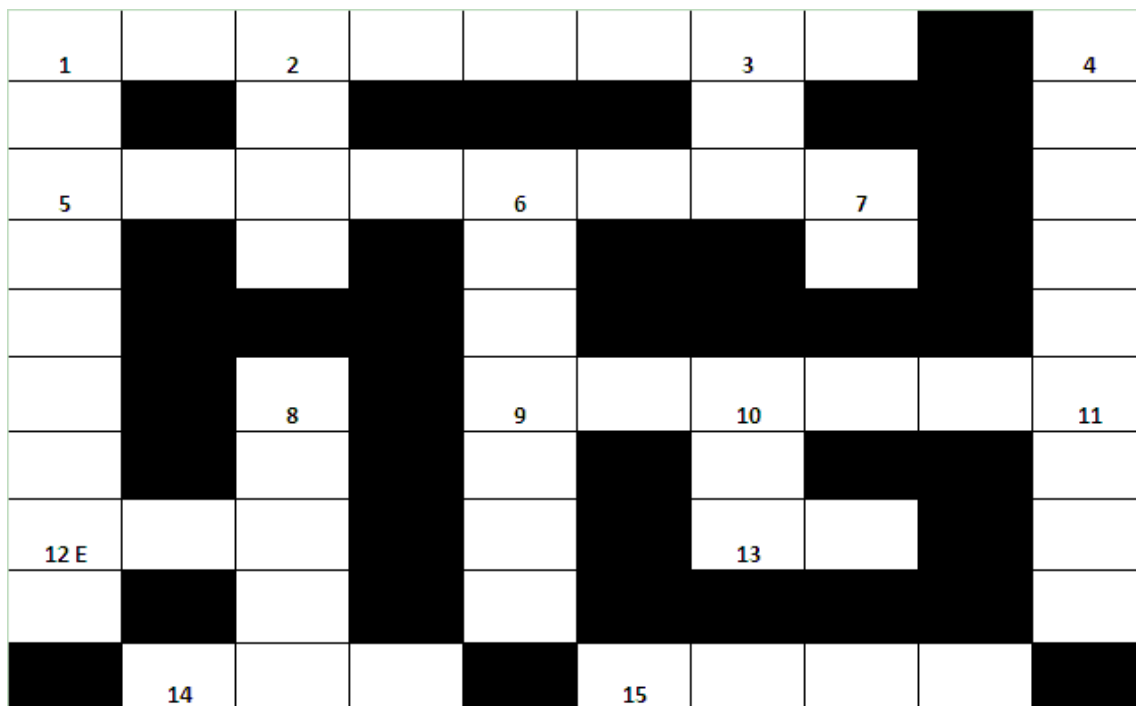
He tweets as @testalways.

You can find some interactive testing puzzles on his website [www.testalways.com](http://www.testalways.com)





# TESTING CROSSWORD



## Horizontal:

1. A software item which is the object of testing (8)
4. Test case selection that is based on an analysis of the internal structure of the component, in first word (5)
5. It is performed to verify that the completed software package functions according to the expectations defined by the requirements / specifications, first word (8)
9. It is a QA tool that is designed to improve workflow, visibility and turnaround time (6)
12. A test case design technique in which the test case suite comprises all combinations of input values for component variable, in short form (2)
13. The short form of web analytics (2)
14. The certification is intended to establish standards for initial qualification and provide direction for the testing function through an aggressive educational program, the first 3 letters in short form (3)
15. Testing conducted at client / customer environment, the first word (4)

## Vertical:

1. The product of PushToTest (9)
2. \_\_\_\_\_ provides for the implementation of compatible keyword-driven test automation framework, in short form (4)
3. Middle words in the term LEARN (3)
6. It is a fully functional, commercial-grade performance testing product (7)
7. It provides mock objects for interfaces by generating them on the fly using Java's proxy mechanism, in short form (2)
8. It is a set of tools for tracking bugs reported by users to a central site (5)
10. An online test case management and test-tracking system built with PHP, in short form (3)
11. Short form of Software Development Life Cycle (4)

# Answers for Last Month's Crossword:

E	B	F		W		B	U	G	
L/N/R			S	A	H	I		I	T
Q	U	N	I	T		N		S	
	T		K	I	R	A	N		
	E		U	R		R		S	T
	S	E/I	L			Y		Y	
	T		I		U			S	
R	R		E		N		B	T	R
	E		E		I			I	
D	E	F	E	C	T	S	U	N	G



*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 ?**



## Our Testimonials

"There is a set of testers from India, Lalitkumar Bhamare and Pratikkumar Patel who are also doing a good job.

They are the Tea-time With Testers. I want to mention and thank them for their efforts.

I sincerely appreciate their work, passion and style."

**- Pradeep Soundararajan**

Director, Co-Founder  
Moolya Software Testing Pvt Ltd.



Dear Pradeep,

Sincere thanks for your kind words and appreciation.

-Editor

To all the people who are associated with this magazine,

A very well written and organized magazine! Keep up the good work. It was really good reading it.

I couldn't move from my place until I finished reading entire magazine!!

Thanks! Keep up the good work!

- Ambika Kulkarni

Hi,

I got to know about this magazine through QT. QT is the best platform for testers to share their knowledge. I wish good luck to both QT & Tea Time With Testers.

-Shivani Shukla

Great ezine !

- Larry Britten

Please let me be the part of **great group** like **Tea-time with Testers**. It is very interesting magazine.

- Xiomara

I recently got chance to read your magazine and it is very impressive.

- Karuna Govathoti

Quite interesting magazine and website. Good Job !

- Hina Saini

I would like your team to learn more about testing. Your magazine helps me a lot.

-Raja

Heard of ezine from one of my colleague. Very impressive initiative ☺

- Anuradha Kondapally

Hi,

This zine looks really helpful for our whole team! Expect more subscriptions from Kollmorgen.

- Sue Haydt

I love this magazine. It's worthy for all testers to read.


- Padmamadhuri Kopparthi

**Tea-time with Testers** is the magazine which every Indian tester should be proud of. Thank you for your efforts.

- Lovina D'suza

# 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](mailto:teatimewithtesters@gmail.com)

# in ne>xt issue

articles by -

A close-up photograph of a metal tag and a small metal clock. The tag is rectangular with rounded ends and has the text "IT'S ALWAYS TEA-TIME" engraved on it in a stylized, hand-drawn font. The tag is attached to a chain. Below the tag is a small, round, metal clock with a visible face and hands. The background is a dark, textured surface.

IT'S  
ALWAYS  
TEA-TIME

Jerry Weinberg

T Ashok

Joel Montvelisky

Michael Larsen

Darren McMillan

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 Image- DreamsTime

## Core Team:

Kavitha Deepak (Bristol, United Kingdom)

Debjani Roy (Didcot, United Kingdom)

Anurag Khode (Nagpur, India)



Anurag



Kavitha

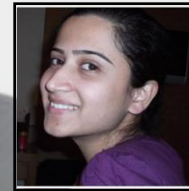


Debjani

## 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



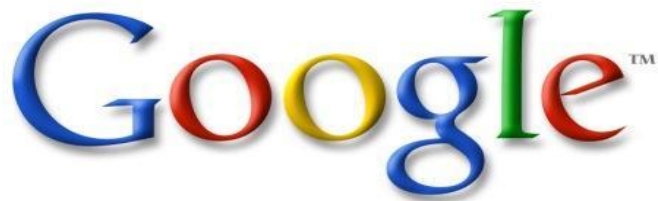
Chris



Gautam

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

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



Join our community on



Follow us on



[www.teatimewithtesters.com](http://www.teatimewithtesters.com)

