

# Tea-time with Testers

APRIL 2012 | YEAR 2 ISSUE III

Jerry Weinberg

Why Congruence is Essential in Management

Paul Carvalho

Exploring in a Financial World

Fiona Charles and Jerry Weinberg

Is Testing a Service?

T Ashok

Properties of a Good Test Scenario/Case

Joel Montvelisky

Why You are NOT a Professional Tester

Bernice Ruhland

Training Someone New to Testing

Samarjeet Mohanty

Software Performance Engineering



Mark Your Calendar  
for  
**LIVE and INTERACTIVE webinar**  
by  
**JAMES BACH**



**Topic : What Testers Need to Learn**

**Date : May 27<sup>th</sup> 2012, 10.00 p.m. [P.S.T.]**

**Stay tuned for Registration and Payment details**

**[ Want to sponsor this event? Write to us on [sales@teatimewithtesters.com](mailto:sales@teatimewithtesters.com) ]**





# TEA-TIME WITH TESTERS

**First Indian Testing Magazine to reach 91 Countries in the world !**

Created and Published by:

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

Editorial and Advertising Enquiries:

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

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 the editors of ***Tea-time with Testers.***

# Editorial

The GOOD news is here !

Hello Readers,

“Testing Skills” has become a buzz word now a days. And if you want to survive in this era of *skilled testers*, you can't really afford to stay behind, can you?

I and my colleagues were discussing about interviews the other day. One of my colleagues is an excellent tester and most of the times he conducts interviews for new team members. The comments that he made caught hold of everyone's attention.

He said, “Despite of so much of advancement in software testing, testers do not know basic things. They are lacking the skill that makes one better tester. I generally reject all the guys who just mug up those definitions and can not even explain basic things. I'd prefer a tester who knows beyond Certification books, practices the craft of testing and develops his own techniques. ”

And I feel that he is absolutely right. As I have always said, Certificates can help you get job but only your testing skills are going to define your growth.

Now you must be wondering , “How do I improve my testing skills? What should I learn in order to test better? How do I decide what is good for me and what is just waste of time?”

Don't panic ! We are here . We have some special plans for all of them who want to make the difference. To start with, we have arranged for Live and Interactive webinar by James Bach where he will teach you where to focus, what to learn and what to avoid.

Stay tuned for more updates. We will get back to you soon with details.

Enjoy this beautiful issue of *Tea-time with Testers* till then.

Sincerely Yours,

- Lalitkumar Bhamare

 editor@teatimewithtesters.com





# QuickLook



## Editorial

### What's making News?

### Tea & Testing with Jerry Weinberg

### Speaking Tester's Mind

Is Testing a Service? - 16

Exploring in a Financial World -20

### In the School of Testing

Training Someone New to Testing - 30

Software Performance Engineering - 34

10 Reasons why you are NOT a professional Tester - 40

## T' Talks

Properties of a Good Test Scenario/Case - 46

## Testing Puzzles

by Sebi



## Crossword

by



### Testing Puzzle – S.T.O.M. Contest

### Our Testimonials

### Family de Tea-time with Testers



# What's making News?

- find out the latest happenings in the technology world

## NelsonHall publishes Software Testing market analysis

**NelsonHall has recently published its latest Software Testing market analysis, and there are some surprises.**

The software testing market has rebounded very strongly after the exit of the 2009 crisis. The scale of the rebound has been unexpected: NelsonHall estimates altogether that the number of software testing specialists i.e. career testers, has grown by 65% in two years, rising to ~170,000 career testers across the world. This growth level is rare in the IT services industry, if not unprecedented.

The report has been constructed following the usual NelsonHall market analysis methodology, multiple in depth vendor interviews with the markets leading protagonists including, Accenture, Amdocs, Atos, Capgemini/Sogeti, Capita Assurance and Testing, CAST Software, Chakkilam Infotech/Cigniti, Cognizant, CSC (including AppLabs), Experimentus, HCL Technologies, HP Enterprise Services, HP Software, IBM Global Services, IBM Rational, Infosys, iGATE/Patni, ITC Infotech, L&T Infotech, Logica, Mahindra Satyam/Tech Mahindra, Maveric Systems, MindTree, MTP, QA Infotech, Smartesting, Sopra, SQS, Steria, TCS, Tieto and Wipro.

The report consists of 89 pages, consists of 9 chapters and 26 data charts and was conducted by NelsonHall's IT Outsourcing research director, Dominique Raviart. If you'd like to find out more about the report and NelsonHall's findings please contact **Rob Hughes**.

## About - NelsonHall

Founded in 1998, NelsonHall is an analyst and advisory firm with an evidence-based approach to market and service provider assessments and an unrivalled BPO and outsourcing knowledge covering an extensive range of business processes and industry sectors.

NelsonHall works closely with its clients to create a value-based relationship using its unrivaled outsourcing knowledge to act as a trusted advisor, providing answers and making business sense of the complexity and challenges faced by both service buyers and service providers within the global outsourcing market.

**Credit: Cisionwire**

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



Are you interested in publishing the news about your own firm, tools, community and conferences in **Tea-time with Testers?**

Then write to us at:

[contact@teatimewithtesters.com](mailto:contact@teatimewithtesters.com)

with “**News Enquiry**” in your subject line.





Announcing...

# Smart Tester of the Month Awards !!!

## ❖ Winners for Testing Crossword :

Saritha S.P (Hitech Outsourcing, Kochi)

Somesh Agrawal ( LogiNEXT Software, Noida)

Devaganaraja Gopalasetty ( Liquid Hub India , Hyderabad)

Vemula Krishna ( Liquid Hub India , Hyderabad)

Akash Patel (Silver Touch Technologies, Ahmadabad)

Siji Kurian (Hitech Outsourcing, Kochi)

Niranjanadevi Muthuvelu (Chennai)

## ❖ Winner for Testing Puzzle :

Sandeep Rao.

**Congratulations !**



**Discussion helps !**

**How about talking with us on Facebook?**

**Come ! Let's have a nice Tea-time there !**

**CLICK ON THE PAGE BELOW TO JOIN US**





Wall  
Info  
Photos  
Links  
Discussions  
WELCOME

About  
A Free Software Testing Magazine and forum to share, discuss, guide/learn an...  
More

202  
like this

9  
talking about this

Likes  
Teach Testing - an  
Create a Page

Tea time with Testers ▶ WELCOME Like

Magazine



"Tea-time with Testers" is a FREE Software Testing Magazine dedicated to all disciplines in the field of Software Testing.

In this magazine you'll get to read variety of articles on software testing & also an opportunity where you can share your own views, win awards, contribute your own ideas, guide others and enjoy every bit of Software Testing.

We are sure , every sip of your tea will be worth enjoying while reading us.



Find us on  
Facebook

# Look Who's Rising



Fifteen Months

11000+ Readers

91 Countries

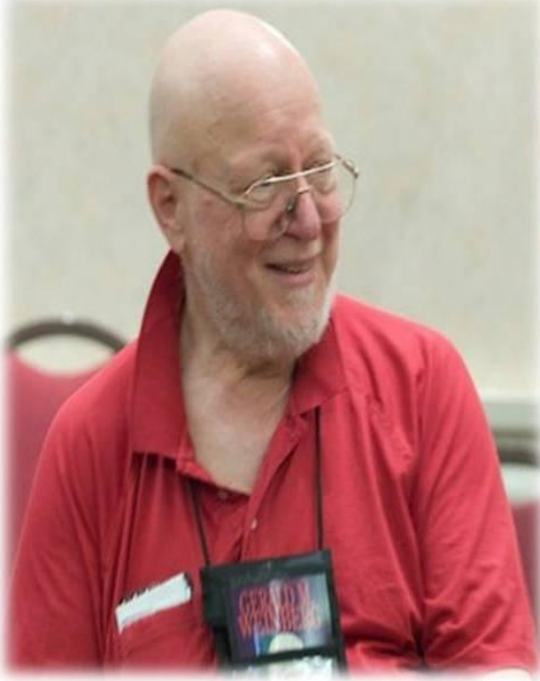
ONE Magazine

## TEA-TIME WITH TESTERS

Subscribe here Right Away to get our all Issues for FREE



# Tea & Testing



with

# Jerry Weinberg

## Why Congruence is Essential for Managing (Part 2)

### 1.4 The Number One Random Process Element

Even though that's not my path, I believe it's extremely important that SEI and other institutions pursue Path 1.

For one thing, it is the path we in computing have always pursued and know best: figure out ways to achieve quality by eliminating people from the equation. Secondly, it has had some phenomenal successes.

I recently read about William Shanks, who in the 19th century took 20 years to compute pi to 707 digits, and made a mistake in the 528th decimal place. I also read about D. H. Lehmer who in the 1930s proved the 257th Mersenne number was prime, using two hours a day for a year.

These two cases can be taken as examples of how tools have helped increased quality and productivity. Today, to compute pi to 707 decimal places, I can invoke Mathematica on my desktop with the program:

N[Pi,707]

Ten seconds later, I have my 707 places, all correct this time.

Or, to determine if  $(2^{257} - 1)$  is prime, I write

PrimeQ[ $2^{257}-1$ ]

Ten seconds later, my Macintosh asserts this is true.

Because of software technology, I have achieved truly spectacular increases in performance, cost, and quality. But these two problems share an important characteristic—they involve essentially no management on my end!

If they did involve management, the spectacular gains made possible by tools would evaporate. What would happen in your organization if a customer submitted a request to compute pi to 707 decimal places. Is this a 10-second job?

I had a number of my students conduct this pi-to-707-decimal-places test in their organizations, secretly asking a customer to submit the request.

Here were some of the results:

- After one week, the request form was returned by a clerk marked, "Incorrectly completed."
- The request form was never returned, never acted on, and had not been heard of again three months later. This was the most frequent "response."
- The customer received a call from a secretary to schedule a meeting with an analyst. The analyst had no available time for more than a month.
- The request form was returned in two days with pi to 10 decimal places pencilled in.
- The request form was returned in ten days marked, "You're not serious!"
- Several replies were in terms of programming estimates, which ranged from 3 weeks to four months.
- Two customers were referred to other customers—one who was using Mathematica and one who was using Maple (which are similar mathematical applications). These other customers graciously solved the problem in a few minutes.
- One customer got a printout of pi to 707 decimal places—actually 1,000 decimal places.
- One customer was asked on the phone whether he wanted a printout or a file on disk. He asked for the file, and was given it by hand less than an hour later.

To me, this silly little survey simply confirms what I have observed directly in dozens of organizations. The variation in service produced by these organizations didn't come from the variation in technology, because all had access to the same technology. The variation came from the variation in management.

In software work today:

Management is the number one random process element.

You don't have to take my word for it, because you can use your own experience. Recall the best software engineering manager you have known versus the worst one. How much did their organizations or projects differ on performance? On cost? On quality?

Now, ask any quality specialist how you go about improving quality, and nine times out of ten you will be told:

1. Identify your number one random process element.
2. Take steps to reduce the randomness in that element.



The problem, of course, is not so simple, because those very managers are the people in charge of changing the process. In other words:

The number one random process element stands in the way of improving all the other random process elements. For some strange reason, many change agencies continue to make the same mistake. They assume the same people who got us into this mess will somehow be the best people to lead us out of it.

All the management tools in the world will not help if the managers are incongruent:

- too self-centered to see and hear what's really happening
- too muddled to understand what should happen
- too fearful to carry out carefully planned actions

When managers are incongruent—not emotionally centered—all of our fancy cybernetic models are full of sound and fury, but signify nothing.

A controller that cannot control itself is worse than no controller at all:

If you cannot manage yourself, you have no business managing others.

The personal effectiveness of people is what integrates all the other components of software engineering management. You'll never get a Pattern 3 organization with Pattern 2 managers. Instead, you start by getting the effective managers, then they lead the others.

## 1.5 The Road Ahead



These observations furnish the overall vision of the task of this volume. One by one, we must remove the obstacles managers face in using full and appropriate variety in their actions.

First, we must address the largest source of management incongruence, which is the way managers are chosen in the first place. As a follow-up to that step, we must address the strongest variety-reducing factor. That factor is low self-esteem that acts to produce behaviors incongruent with what managers know they should do.

Once they leave behind their incongruent coping behaviors, managers still face many obstacles to requisite variety. Unconscious preferences tilt the manager's actions away from the logic of the job. We must expose the most common of these preferences: personality, temperament, culture, gender, age, modes of perception, and physical capacity. Then we must learn what a manager can do to make them conscious. Consciousness, however, is not sufficient to change actions. We must also examine how people become addicted to their destructive behaviors. Then we have to learn how much more than logic or morality we'll need to overcome addictions.

Finally, we must examine the managers' relationships to the most effective variety-producing tool at their disposal —other people. We need to examine how managers can improve their interactions with individual workers, colleagues, and bosses. More than that, we need to explore the special relationships that exist between managers and teams.

That's a big menu, but it won't help at all if you don't have lots of practice. The sooner we get on with the reading, the sooner you'll be able to go into action.

*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

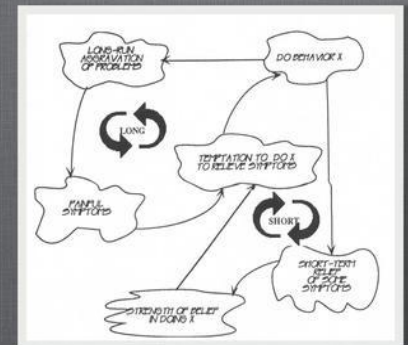
**MANAGING YOURSELF AND OTHERS** is yet another famous book written by Jerry.

Becoming an effective manager is the subject of this volume in Gerald M. Weinberg's highly acclaimed series, *Quality Software*. To be effective, managers must act congruently. Managers must not only understand the concepts of good software engineering, but also translate them into their own practices. Read this book to find out more.

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

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

## MANAGING YOURSELF & OTHERS



GERALD M. WEINBERG

**TTWT Rating:** ★★★★★



A photograph of a green, conical pendulum bob hanging from a thin wire. The bob is positioned over a light-colored sand surface. In the sand, there is a faint, circular mandala-like pattern. The entire image is framed by a dark blue border.

# Speaking Tester's Mind

- straight from the author's desk

# Is Testing a Service ?



by **Fiona Charles  
& Jerry Weinberg**

Remember those stories of ancient India when Rulers used to conduct **Vidwat Charcha**?

Vidwan means **expert** or Pandit , Vidwat means "**of experts**" and Charcha means **conversation**.



Yes, to quench their thirst of knowledge or to understand different aspects of subject they used to conduct those Vidwat Charchas and just by listening to the conversation, discussion between two experts; Rulers used to gain knowledge.

They did not always find answers in epics and some times their interpretation of what they read left them confused. But, listening to the conversation between two experts did clear their every doubt.

Well, we (Tea-time with Testers) are no Rulers but we firmly believe that conversation between two experts can give us **that** knowledge which is hard to get otherwise.

We are glad to publish this article which arose out of an email conversation between **Jerry Weinberg** and **Fiona Charles**. Fiona had just participated in POST, the Calgary "Perspectives on Software Testing" peer conference.

Find out what knowledge you gain out of this Vidwat Charcha.

- **Editor**



### **Fiona Charles:**

I just spent the weekend at POST, the Calgary peer conference Lynn McKee and Nancy Kelln founded after coming to TWST. It was the third POST, and the first one I've been able to make it to. It was just great, though I think I upset the applecart a little with my presentation. The topic was "testing as a service". This comes from *Lessons Learned in Software Testing*, where Cem Kaner, James Bach, and Brett Petticord wrote that testing is a service to the project. It's a fashionable stance for context-driven testers. I argued that it's a very dangerous thing to say from a practical standpoint, because it in fact puts testing outside the project. Doing that emphasizes our second-class status and can even lead to a belief that the testing service is optional. Testing may be appropriately optional in some contexts, but in most cases I think separating testing from "the project" does more harm than good.

I prefer to say rather that testing is an integral part of software development, and that software development is a service to its business stakeholders.

Of course, there are other ways in which you can see testing as a service, e.g., if you are selling testing services, as I did for many years and do now to some extent. I talked a little about that, too.

### **Jerry Weinberg:**

I would say testing is a service in the same sense that, say, my team of physicians did a service for me during my cancer period.

In that case, the service-providers are higher-status than the patient. Could be that way in testing business, if we wanted to make it that.

Or maybe we want it to be a partnership, like a couple happily married for 50 years.

### **Fiona Charles:**

I want the whole project to be a partnership.

I don't have a problem with "testing is a service". I do have a problem with "testing is a service to the project". To me, that separates testing from the project—unless project management, business analysis and programming are also a service to the project. They may well be, but the people who do those things don't describe their work that way.

### **Jerry Weinberg:**

It can be either way. To take a less controversial example, providing network access and capacity could be within a project or a service to a project. For testing, there are pluses and minuses for each way.

Some people believe testing (at least some) should not be in a position to be influenced by project management.

Other people see testing ideally as a cooperating function, fully integrated with the rest of the project all of the time.

In general, I prefer the second, but not if management is not capable of leaving them alone to do their job, not pressuring to give the answers that make the managers look good.

**Fiona Charles:**

In my experience, management that pressures testers to give rosy answers (i.e., lies) will do that whether or not testing is integrated with the project team. It can be helpful for testers to have a separate reporting line, but when organizational cultures are rotten the rot usually starts above the level in the hierarchy where the development and testing lines diverge.

**Jerry Weinberg:**

You're right about that, yet though the rot develops there, it's usually easiest to detect early at the tester/developer level.

Anyway, I'd rather not work with organizations where I have to dig behind screens to see the rot.

**Fiona Charles:**

The idea of tester independence has come full circle in the course of my career. Early on, we fought hard for organizational independence—not primarily because of pressure to make people look good, but because managers had programming backgrounds and tended not to understand or care about testing, meaning they often didn't allow us to test enough or well. I've frequently had to deal with idiocies like "your testers are asking too many questions and slowing down development" or that old song "your testers are finding too many (or the wrong) bugs".

**Jerry Weinberg:**

Yep. My career, OTOH, started earlier, when the idea of separating testing from development basically didn't occur to us. Testing was simply part of the software-building job, though we might have members of the team specializing in testing for a particularly difficult-to-test part of the project. But they were always under the single project manager, one way or another.

**Fiona Charles:**

One problem (or at least a perceived problem) was that once testers achieved organizational independence, the separation often hardened into structural antagonism. This was reinforced by the counter-productive idea—promoted by both managers and testers—that testers were gatekeepers. Among other detrimental effects, many testers became judgemental, burning with a righteous belief that only they cared about quality. I've always challenged this in my teams and worked to promote cooperation, but it seems to have pervaded many organizations. On the big project I worked on in the UK, our government customer was convinced that it would create a conflict of interest if my test team worked closely with programmers.

**Jerry Weinberg:**

<sigh> We've known that scenario several times.

**Fiona Charles:**

The push to integrate testers more fully into project teams has come partly from agile, with its emphasis on collaboration, and partly from people like me who have fought for decades to get testers involved earlier in projects when they could really contribute to getting better requirements and more testable code, as well as getting their hands on code to start finding bugs earlier.

**Jerry Weinberg:**

We had already won that battle in 1960 or even earlier, without realizing we were in a battle.

**Fiona Charles:**

I don't believe that managers are any smarter about testing than they ever were, but I also don't believe the organizational separation has worked very well to alter their behaviour. And it has certainly contributed to bad behaviour among testers and bad feeling about testers among programmers. I don't know any easy answers, but I do think testers have to be tough as well as skilled. We need to maintain independence of mind while collaborating fully on software projects.

**Jerry Weinberg:**

I can attest to that. (BTW, if we do collaborate on an article, we're going to have to battle over the spelling of "behavio(u)r.")

And testers have to be skilled at reframing, but be careful you don't put all the burden of change on testers. That's been part of the battle all along.

Still, all we can do as testers is change what we can do, and hope that others will follow. That does happen (sometimes).



**Jerry Weinberg** has been practicing, teaching, lecturing, consulting, coaching and writing about software programming and testing since the 1950s. With decades of experience and accumulated knowledge he's written more than 80 books and has dedicated his life to helping others be the best testers they can be – despite ever changing testing trends.

Jerry recently got interviewed by **uTest**.

After reading this "[Testing The Limits](#)" interview you'll find out the biggest lessons Jerry's learned over the years, how his books remain top sellers 20 years after their release and what the biggest issues facing testers today are.

To keep up with Jerry visit his [website](#) or follow him on [Twitter](#).



**Fiona Charles** teaches organizations to match their software testing to their business risks and opportunities. With 30+ years experience in software development and integration, she has managed testing and consulted on testing on many challenging projects for clients in retail, banking, financial services, health care, telecommunications and emergency services.

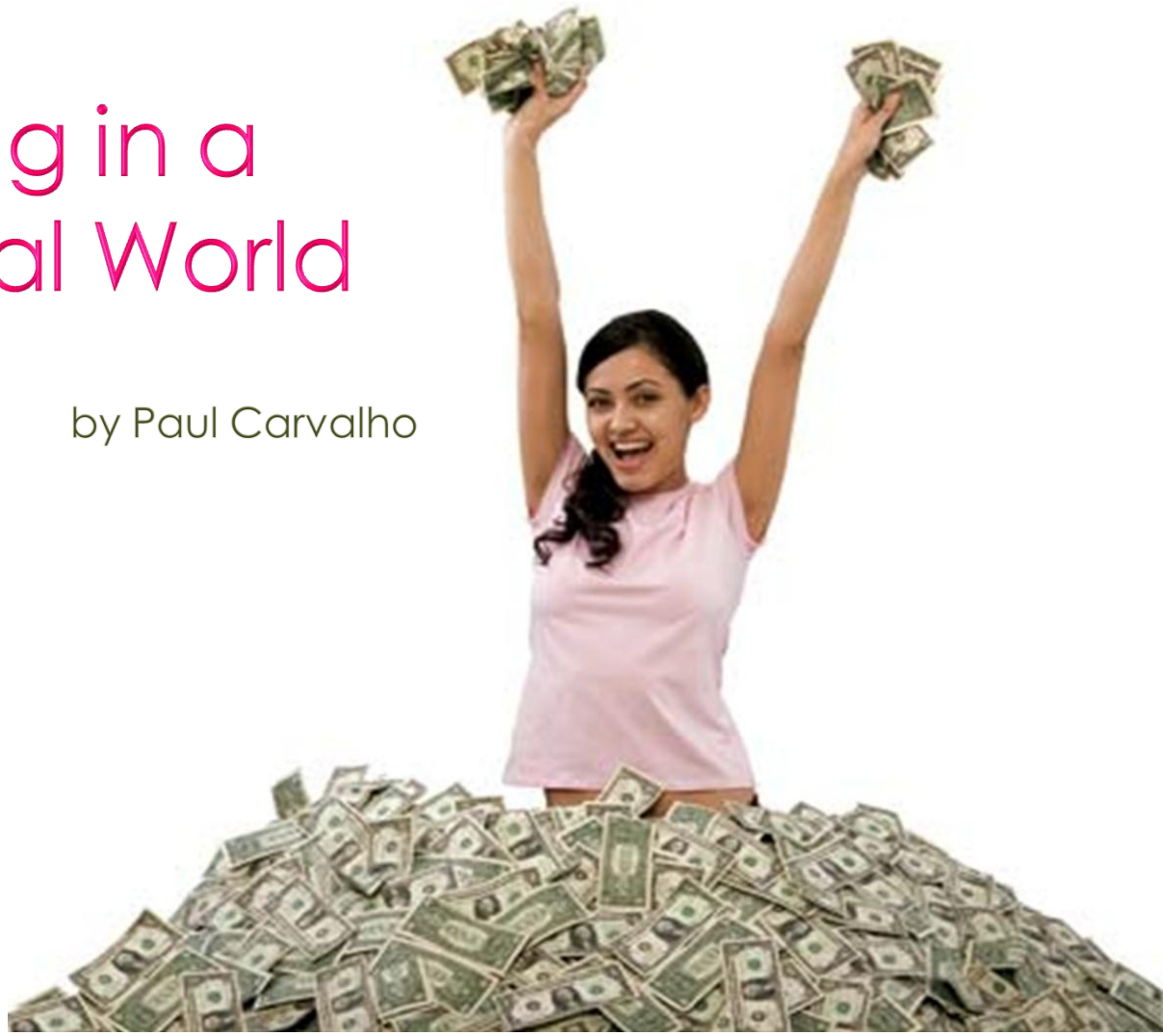
Throughout her career Fiona has advocated, designed, implemented, and taught pragmatic and humane practices to deliver software worth having—in even the most difficult project circumstances. Her articles on testing and test management appear frequently and she speaks and conducts experiential workshops at conferences. Fiona edited *The Gift of Time*, and guest-edited —Women of Influence—, the January 2010 special issue of Software Test & Performance magazine.

Fiona is co-founder and host of the Toronto Workshop on Software Testing.



# Exploring in a Financial World

by Paul Carvalho



I looked at the auditor's questions in the email again and blinked. "Do you have Test Plans? Do you have Test Cases?" If the auditor expected traditional test documentation, then the answers to both questions are "no." I don't think they will like those answers though.

I got up and walked over to talk to my boss. I asked him if it was worth it for me to write a lengthy response to describe how we test or if was possible to explain it in person. He said that an in-person conversation would be better and that he would arrange a meeting. A major contract with a big bank was riding on the outcome of this supplier audit of our development practices. We made financial analysis software for financial institutions and this was not the first time we had been asked to elaborate on our non-traditional testing practices to a large bank.

A week later, we had two visitors from the bank at our company: the project manager who was championing the integration of our software into their system, and a lead auditor interested in the details of our approach to Quality. I had two hours to win them over or potentially face the reality of losing a major bank as a client. No pressure!

I began by clarifying the objective of the meeting: *to explain how we test in a way that meets or exceeds their expectations*. I referred back to the audit questions and said that although the answers to their questions were "no", it doesn't mean that we don't test. We just happen to test in a way that is different from what the questions asked. To use an analogy, the questions were like asking if I had

driven a car to work. If the answer happens to be “no,” it doesn’t mean that I didn’t use some other form of reliable transportation.

I then offered a disclaimer and said that I would not discuss *all* aspects of how we (as an organisation) try to “assure” Quality in our projects. A conversation like that would include people and processes well beyond the scope of my responsibilities as Test Lead. That is, a holistic discussion of “Quality” would include things like management meetings and decisions, requirements management, technical design and architecture, reviews (requirements, design, and code), source control used for configuration management, project management, unit testing, deployment practices, and so on. The testing practices that I planned to speak about were one part of the bigger picture, and our team is not solely responsible for Quality Assurance.

Our guests were okay with this. They had already asked other development leads about many of the items I mentioned and were curious to know more about how we tested in development. They were not used to getting “No’s” in response to the Testing-related questions.

Rather than jumping in and showing them some of our supporting test documentation, I said that I planned to explain some of the context and reasons for choosing a different approach first. It would help them understand what they were looking at and why. We will look at some of the documentation afterwards and conclude with a discussion of whether or not it was sufficient for their needs.

## Challenges in Good Testing

To set the stage, I highlighted some challenges in software development, and software testing in particular. Some of the points I covered included:

- What is Quality?
- Communication
- Test coverage
- Inattentional Blindness
- What is a bug?

I covered the first few points with open-ended questions. I hoped to convey that these are complex ideas that mean different things to different people. I spent some extra time on the last two items since they related directly to our choice in testing approach and the corresponding lack of test case documents.

For Inattentional Blindness, I showed a YouTube video to demonstrate this effect and followed it up with a simple definition: *sometimes when you focus on one thing, you miss something else in plain sight*. There are many examples of this in everyday life.

I explained how this affects testers when testing with traditional ‘scripted’ test cases. When a person follows detailed steps in a Test Case, they miss important system behaviour because their attention is focussed on following the test steps. A good tester needs to be aware of this effect and considers ways to reduce the chance of missing important information while testing.

For us, on our test team, it meant that we did away with the detailed test cases and focussed on the higher-level test strategies, models and techniques that are used to generate them. We also employed monitoring tools, automation scripts, and pairing with other team members when we tested to try to reduce this effect.

Regarding the definition of a “bug”, I asked our visitors how they would know if something was a bug when testing. The pass/fail “expected behaviour” associated with test cases is not as simple or trivial as we are led to believe. There are sometimes competing ideas for what the “right” system behaviour should be. For example, if the application matches the documented specifications but doesn’t match the user expectations, is there a problem? How would you check that? How might you even recognize that kind of discrepancy?

It is very difficult to predict in advance what the expected system behaviour should be without learning something about the system and speaking or checking with other project stakeholders. On our test team, we figure that out in real-time using *oracles*. An oracle is a way to tell if you have a potential problem. This may be a person (e.g. a Subject-Matter Expert), a reference of some kind (e.g. a requirement or specification), or something else. One important catch is that oracles are heuristic; they are like guidelines and are sometimes wrong. Since they don’t always give you the right answer, you need to use judgement when applying them.

At this point, I could tell by the looks on our guests faces that I needed to pause and check-in with what I had covered so far. They said that the ideas of oracles and inattentional blindness were new to them and they found them interesting. They were also excited to learn how we adjusted our testing practices in innovative ways because of these ideas. So far, so good.

## Testing to Fill Gaps

Good testing complements the way the rest of the development team works. For the most part, our development team followed an iterative Waterfall approach with some Agile/Scrum practices mixed in. Requirements had been known to change and evolve during projects. The mandate for our test team was to provide additional insight beyond the functional checks and verification activities already performed by other developers. Given our short development cycles, we also needed to provide good information quickly.

The most efficient way I know how to do this is to leverage human ingenuity via an Exploratory Testing (ET) approach. I gave a quick description of ET and explained how it is a structured approach to real-time learning, test design, execution, and analysis of the results. Every thinking human being follows a similar process at one time or another when solving problems, and our team applied this approach to test software in a thoughtful, intentional manner. ET is not accidental, random or haphazard testing; it is done with purpose.

When working on a development project, we also performed regular risk assessments as a team to ensure that we focus and act upon current priorities and information. This allowed us to adapt to changing requirements and needs as the project developed. From there we create multidimensional Testing Strategies to provide different views of the SUT. We store these Test Strategies in our internal Wiki, and I explained to our guests how these would be the closest things to formal “Test Plans” that we had. The Wiki was version-controlled (so you can view the audit history of every change made to the content), had collaboration elements, hyperlinks to other pages (e.g. for traceability to requirements), and contained all the important information of *what* we were going to test, *how*, *who*, *where*, and *why*. The only thing missing was the project management (“*when*”) part of the plan (e.g. milestones, schedules, etc) and I explained how our boss managed that separately.

When it came to the hands-on testing activity, all of the testers on the team were trained Exploratory Testers. I trained them myself. Each tester knew how to design tests, execute, observe, communicate, document, and share what they had learned. We used a Session-Based Testing framework to provide accountability to the testing as well as to facilitate communication, understanding, and knowledge-transfer within our team.



Session-Based Testing (SBT) is a wrapper around the testing activity that provides some key benefits:

- clarity of purpose for the testing activity
- uninterrupted time-box (e.g. perhaps an hour or two) to allow the tester to focus and complete a significant effort of work
- standard way to keep records of the testing performed
- a “debrief” activity with the tester

I explained that SBT may be used with many testing approaches (not just ET) and that it may replace the traditional Test Documentation often performed by testing teams. Since one of the main reasons for this client visit was to understand the “No Test Cases” response, I spent some time discussing our SBT/ET records.

We kept notes of the testing performed in simple text files, stored them in a common repository, and reviewed/debriefed them regularly. At the end of each project, we moved them into our Version Control System next to the application source code.

At this point, I brought up some sample test session records to walk through the testing described within them. I explained that it takes some practice to learn to write good, clear testing notes, and that good, regular feedback is an important part of the learning process. Each member of our test team could review any other tester’s notes.

The Test notes are written primarily to a testing audience, so I don’t worry about everyone outside the test team understanding everything contained within the notes. This is similar to reviewing a programmer’s source code - there is a barrier to understanding a programming language that becomes clearer with practice and competence.

The sample test session record had a clear ‘Charter’ (mission, purpose, or objective), derived from the Test Strategy kept on our Wiki. The Test Notes section identified oracles such as references to requirements, designs, and people that would be helpful for both understanding scope and identifying potential problems. What followed was a description of the tester’s learning path in identifying and exploring potential risks associated with the feature under test. Descriptions of test techniques and heuristics were interspersed with industry terminology and system features. A few simple tables and checklists hinted at some of the complexity in the paths explored during this particular session. There were also a few “bugs” found and recorded in our bug tracking system. The final sentence in the Notes section wraps up the test session by referring back to the test charter to indicate whether it was completed or if there were important risks still to investigate.

## Completing the Meeting Charter

With the test notes displayed on the screen for the guests to see, I returned to the reason for the visit: “No Test Cases.” I explained how the tester in this particular session had executed dozens of test ideas and documented them at a level at which we could understand what was done. Any tester on our team would be able to review these notes and reproduce the results with a high level of confidence. More importantly, in addition to several key observations, the notes captured pieces of conversation with other project team members that provided insights into the feature under test not documented elsewhere.

The Test session notes went above and beyond what I understand traditional test documentation could accomplish. These records contained information on multiple levels that I couldn’t think of a more efficient way of capturing.

At this point, I turned to our visiting guests and noticed the amazed and excited expressions on their faces. I asked, "Now do you understand why I said we have no test plans and no test cases? Will this do?"

They both laughed! With broad grins, they both agreed that our test documentation and processes were more than enough for their needs. They were pleased with how seriously we took our testing. They thanked me for the presentation and insights and said that they would rephrase the contract's testing questions when they returned to the office. Jokingly, the auditor asked if I could give a similar presentation at her office so their development team could learn from how we tested. The mood was certainly happier leaving than when we started the meeting.

I thanked our guests for taking the time to learn more about how we tested our financial software.

We gained a new client that day.



**Paul Carvalho** is a Quality Architect, testing consultant, trainer and coach. He has worked in the Software and IT Industry for over twenty years. Paul helps software companies overcome development inertia and become world-class leaders in delivering quality and value. He specializes in systems analysis, collaborative problem solving, test design and exploratory testing, and has a gift of teaching for understanding.

Paul is an active participant in the professional software development community. He speaks at conferences, blogs, loves scripting with Ruby, and has written articles for several publications including Better Software magazine and Agile Journal.

You can find more of his work on his web site ( <http://staqs.com/> ).

Paul can be reached at [paul@staqs.com](mailto:paul@staqs.com) or on twitter @can\_test

Doesn't that inspire you to share your own success story?

Don't be shy ! Take a paper and pen it down.

We are very much here !

If you have any questions, refer to our Article Submission FAQs or email us at [editor@teatimewithtesters.com](mailto:editor@teatimewithtesters.com) .



*A brand new section....*

# *“The Guiding Star”*



*Coming soon ....*



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.



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

# In the school of Testing

*for your better learning & sharing experience*

# Certifications might get you job but **ONLY** your Testing Skills will define your **GROWTH** !



Don't waste your time in mugging those definitions.

Enroll to our unique online training course where you will learn the real flavour of software testing.

**Announcing special WEEKEND batch on request.**

Find the revised schedule below...

## COURSE SCHEDULE

**Batch starts on 5th May 2012**

**Timing: 10AM to 1PM IST**

**Course Duration: 5th May to 10th June (Weekends Only)**

**Class Duration: 3hr (5min. break each hour)**

## FEE STRUCTURE

**\$ 145.99 USD** for entire course.

Special discount of \$20 for **Tea-time with Testers'** subscribed readers only.

## REGISTRATION

To register for this course , click [HERE](#)

## ABOUT THE COURSE



**~ SOFTWARE TESTING ~**  
**BASIC & ADVANCED**

## Software Testing : Basic and Advanced

Our online 1.5 month testing course develops tester's skills.

It will help you to design your own methods to solve any given testing problem.

Check out the [syllabus](#) [HERE](#)

## BROUGHT TO YOU BY







~ PLEASE NOTE OUR NEW CONTACT DETAILS ~

➡ For Editorial enquiries, write to us on [editor@teatimewithtesters.com](mailto:editor@teatimewithtesters.com)

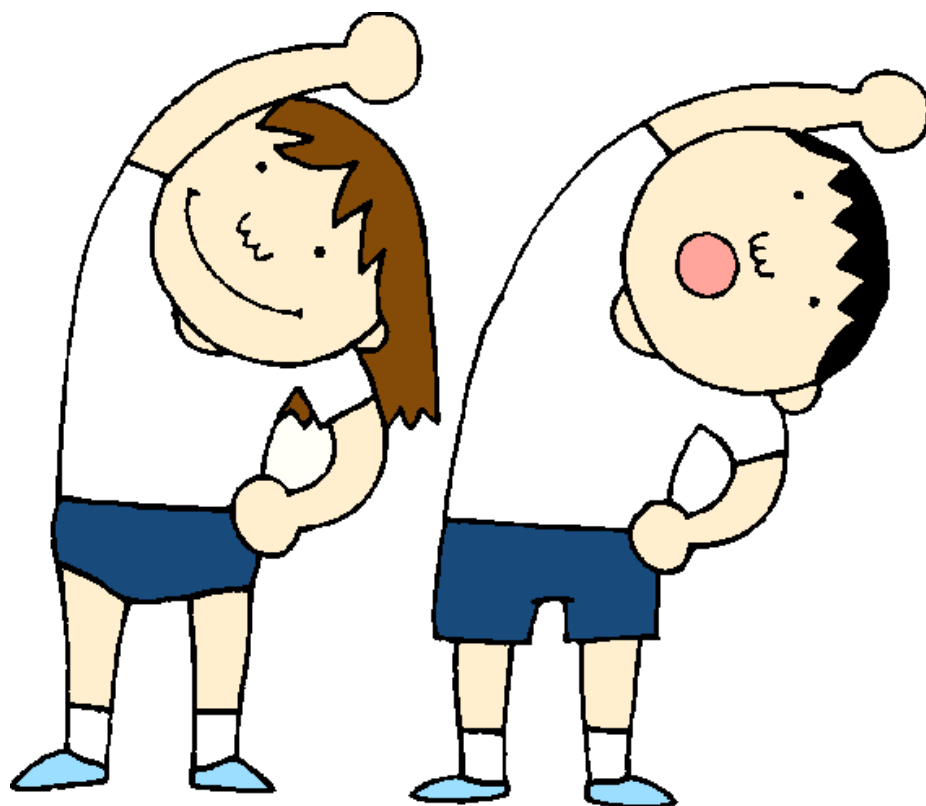
➡ To contact us, mail us at [contact@teatimewithtesters.com](mailto:contact@teatimewithtesters.com)



“Career Development and Learning Strategies for Testers” is a series of articles providing different approaches to develop testers’ skills and knowledge from both a managerial and tester perspective.

*By Bernice Niel Ruhland*

## Training someone new to testing – Part 2



"Career Development and Learning Strategies for Testers" is a series of articles providing different approaches to develop testers' skills and knowledge from both a managerial and tester perspective. My article "Onboarding New Testers" in Tea-time with Testers November 2011 issue discusses how to put together an onboarding program to transition a new employee into your department.

The focus of this two-part article is guidelines for training someone who is new to software testing. Part 1 discussed identifying different sources of training. Part 2 focuses on the actual hands-on training. The techniques and tips provided can be helpful if you are training a new tester or if you are the newbie!

### **Hands on testing**

One of the best ways to learn to test is to actually use the product. Provide any learning material such as the training manual, requirements, and if possible have someone provide a product training session. Identify easier functionality for initial testing which might be following the training manual instructions in the test environment. Walking through basic functionality is a good way to introduce the tester to the product and reviewing if the expected behavior is correct. At this point, the tester is not concerned with identifying testing ideas.

### **Writing testing ideas**

Once the tester has an initial introduction to the product and general testing, it is time to progress to writing testing ideas. A great place to start is with easy bug fixes. You can either identify bug fixes that have been tested or new ones that are in the testing queue. Consider providing guidelines to get the tester started in thinking through testing but do not constrain his thinking with rules and regulations.

For his first bug you may identify testing ideas and walk him through the testing. Then allow him to write testing ideas for the next bug. Review the testing ideas to provide input on additional testing and how you identified those tests. Allow this process to evolve where the tester is identifying initial testing ideas and then expanding upon them during testing. Continue this process until you are comfortable transitioning him to more complex bugs and new functionality. Identify a project that he can be involved during the initial testing stages. This allows him to build relationships with his team members and learn the project at the start.

### **General guidelines to writing testing ideas:**

Provide the tester with general guidelines to get started based upon your testing methodology and environment. Caution! Do not present this as a checklist of things to do and once all boxes are checked you have completed testing. This is not a checklist since testing goes beyond checking boxes of items to test.

Discuss the guidelines with the tester explaining their purpose and limitations. You should be spending sufficient time with the tester based upon your assessment of testing and problem solving experience as well as his progress. Do not use a guidelines document as a substitute for conversations.

*Below are guidelines to get started testing.*

1. Begin by reading all documentation for the problem ID, asking any questions on expected behavior and outcome.
2. Go to a test environment that has production code to reproduce the problem. This provides an initial understanding of the problem to help write testing ideas.



3. Identify positive tests to show that the feature does what it is suppose to do. For example, if you are testing a date field, you should be able to enter a valid date, save the date, and retrieve it.
4. Identify negative tests to show that the feature does not do anything that it is not supposed to do. Testing can include: correct error handling, data validation, and recovery functionality. For example, entering an invalid date produces an error message. Closing the error message allows the user to correct the invalid date and then proceed with positive testing.
5. Perform boundary testing if appropriate. For example, if the field accepts 10 - 15 then test 10 and 15 as a positive test as the field should accept those numbers. Test 9 and 16 as a negative test as the field should *not* accept those numbers.
6. There are other testing techniques such as a test matrix and all pairs approach that helps identify testing samples.
7. Based upon what you are learning during testing, ask yourself the following questions.
  - a. "What else can I test?"
  - b. "What am I missing?"
  - c. "Who can help me better understand the testing problem and/or expectations".
8. It may be impossible to perform all testing ideas. It is important to balance the problem ID's risk level against time constraints and other assignments to determine how much time to allocate. When in doubt, review with the testing manager or testing lead how much time is needed and what level of testing to perform.
9. When making an assumption that the expected outcome is correct; stop to evaluate what evidence is available to prove that claim. Assumptions can be dangerous and can lead to missing important problems.

## **Coaching and conversations**

A training program should be customized to the tester evolving with his progress. Do not create a training program expecting it will not change. This should only be a starting point. Some testers will progress more quickly or slowly through a program. It is important to adapt it to his learning needs.

More important than a great plan, is that you spend time with the new tester. Walk through testing scenarios with him to train and guide. Based upon his progress allow him to take more responsibility for testing bugs with your oversight. Remember just because someone does not ask questions it does not mean they understand. They are new to testing and you cannot ask questions on what you do not know. Your role as a coach is to identify those areas and bridge the gaps. Provide both constructive feedback and positive feedback on progress.

Consider when building a home how important it is to build a solid foundation. A newbie tester also needs a strong foundation in fundamental testing skills, questioning assumptions, and communicating information before proceeding to more difficult testing. Being his testing coach will help him form that foundation.

## Tips for Managers:

- Training programs should be customized to the new tester based upon informal software testing and problem-solving experience.
- Provide learning materials such as books, articles, webinars at different increments based upon the tester's progress and what he is testing.
- Provide general guidelines to help the tester identify testing ideas. Do not provide a checklist approach as testing cannot be measured by checking off a series of to do items.
- Spend time with the new tester explaining how to test by working through testing assignments. Do not substitute guideline documents or reading material for conversations.

## Tips for Testers:

- Read and re-read training material to understand the purpose and how to test. Initially read to understand concepts and gain an appreciation of testing. Once you have hands-on testing experience, re-read relevant material to further understand concepts and how to apply them.
- Connect with other testers through local testing groups and social media to discuss testing approaches. Go beyond what is provided by your manager to learn other testing techniques.
- Read an article or blog every day. Never stop learning from other testers. Also read non-testing articles and blogs as they can help with critical thinking and problem-solving.

## Conclusion:

Training an employee who is new to software testing is different than an experienced tester. Prior to developing a training program, understand his background in problem-solving and testing. Many people have informal testing experience relating to a hobby or job responsibilities not focused on software testing. Gradually introduce different training approaches through reading material and webinars. Start the training process by having him use the company's product and testing easier bug fixes. As his experience increases introduce more difficult bugs and new functionality. As his manager or test lead, you are also his coach. It is critical that you spend sufficient time training and working side-by-side with the new tester. Providing written guidelines and procedures is helpful but should never be a substitute for conversations. Embrace the opportunity to open the rewarding world of software testing to a new tester!

**Bernice Niel Ruhland** is a Software Testing Manager for a software development company with more than 20-years experience in testing strategies and execution; developing testing frameworks; performing data validation; and financial programming. To complete her Masters in Strategic Leadership, she conducted a research project on career development and onboarding strategies. She uses social media to connect with other testers to understand the testing approaches adopted by them to challenge her own testing skills and approaches.

The opinions of this article are her own and not reflective of the company she is employed with.

Bernice can be reached at:

LinkedIn: <http://www.linkedin.com/in/bernicenielruhland>  
Twitter: bruhland2000  
G+ and Facebook: Bernice Niel Ruhland



# Software Performance Engineering :



‘So many types of tests – Confused !!’

*by Samajeet Mohanty*

In this abstract, I’m going to talk about the *Implementation Phase* of SPE where the test scenarios identified, as feasible and critical for a product, are formally tested as part of Performance Test Cycle. Now, before you ask what happened to the phases prior to **Implementation** like **Design** and **Analysis** where a Performance Engineer carries out tasks like “*Workload and Transaction Processing Estimation*”, then don’t worry. I wanted to get the comparatively easy part out of the way first. Hence, this article.

Coming back to the point, I’m sure you would have heard about various types of performance tests being done in the woods. With so many test type definitions laying out there, the work is already cut out for us. On top of that, there are often contradictory explanations for the same type of test. Now, I do not claim to be a so called expert in Performance Testing arena to certify which definition is ironclad or which is incorrect. I believe every application is different and complex in its own way. So as long as you understand your product and the working / boundary test conditions, you can do the required performance test and call it whatever you want.

Having said that, the last thing I want this paper to be is like any other article out there with plain word-by-word definitions. Instead I’ll try to amend a simple pictorial description of the tests as I’ve understood and encountered in my work experience. These were considered a priority preceding any other performance test type. As some wise being said, pictures speak louder than words. Don’t they.

For simplicity, let’s consider an online application which has an Application Server layer (with embedded Web Server functionality) and a Database Layer.



The working conditions for AUT are:

- On a business day (8am-5pm EST) users interact with the application's GUI via Internet, say from multiple geographies.
- The application has EOD processing scheduled via batch/cron jobs off-business hours, say from 9pm-11pm.

Please note that, in addition to above, there might be other **types** of communication (like xml message processing, FTP and so on) between the application components that you will need to simulate to test End-2-End performance. However for current topic in discussion, there are 3 tests that one should compulsorily conduct while testing for performance.

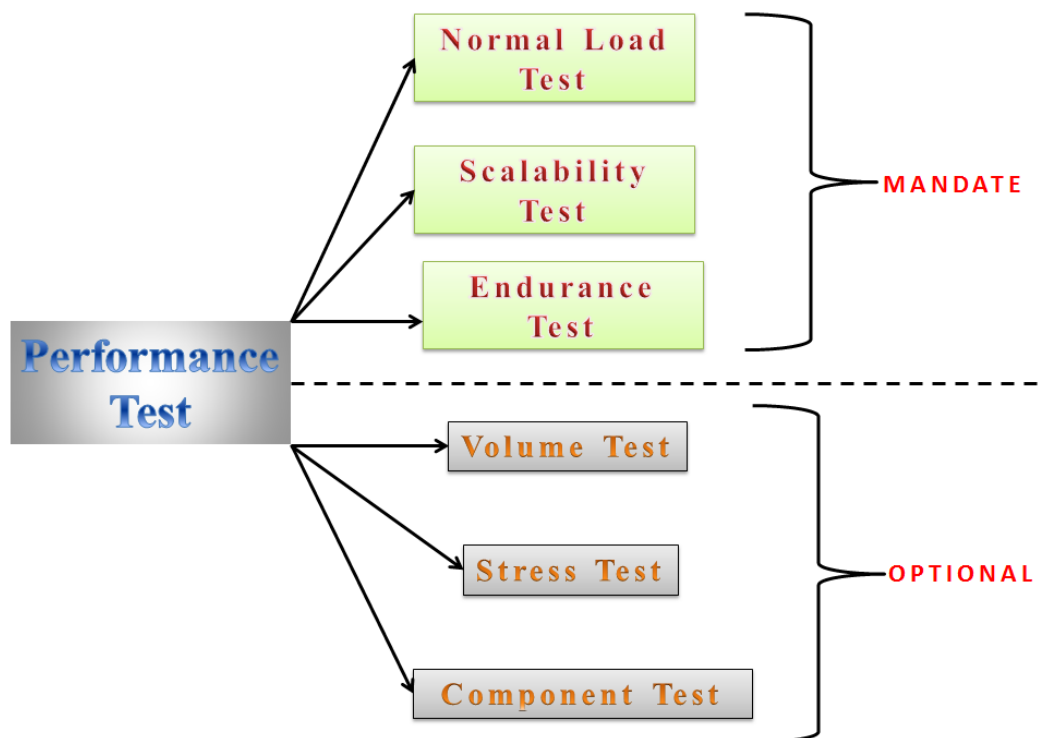


Figure 1.0

Before doing any tests, make sure to be aware of the configuration and tunable for your test environment. These include information like –

- Type of Operating System (with version) on the host server
- Number of CPU's allocated – Physical or Virtual
- Amount of Memory allocated – Dedicated or Shared
- Type and amount of Disk space or Storage available
- Network Bandwidth in use

- List of other applications using the environment at what times
- Garbage collection mechanism, Thread Count, Process Count, Connection pool size
- Test environment to be an exact replica (in terms of at least hardware and software) of production. This however is an ideal scenario which rarely happens and often a scaled down version of Prod is used as QA or test environment. In such a case, project stakeholders should be ready to extrapolate test results of scaled-down environment into Prod assumptions. Remember too much data extrapolation isn't good either. I'll try to touch upon this point in future columns.

Talking to a system admin or architect, who maintains these servers, will provide you with useful insight into the boundary conditions your test is being performed under.

To explain the test types in a better way, let's say the AUT mentioned earlier is a brand new application which is being launched in multiple geographies.

- For initial 6 months, 500 online users are expected to use my application.
- Day 0 database volume would be 100GB
- There's 1 web server (1 CPU), 1 application server (2 CPUs) and 1 database server (2 CPUs). All server CPUs are Intel Core 2 Duo type.
- A dedicated heap memory of 512MB:1024MB (Xms:Xmx) for App server and an SGA of 2GB for DB server.

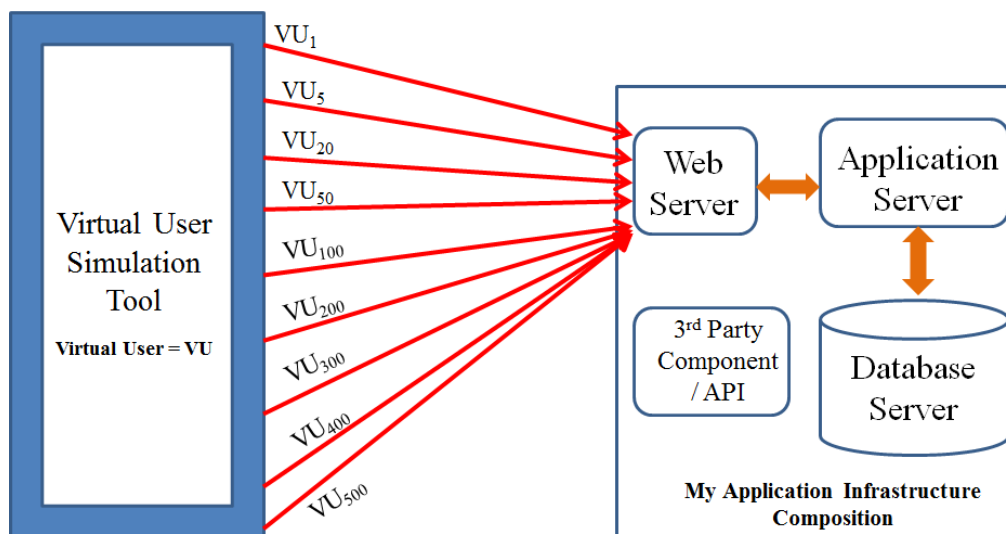
I won't go into too much technical details of remaining tunables configured as they are out of scope for article in discussion here.

## Load Test:

**GOAL:** Find out how the application behaves when expected load is simulated on SUT under normal conditions.

### WHAT DOES THIS MEAN ?

Keep the environment tunables configured as is and simulate a virtual load of 500 users doing day-to-day operations, for say 1 hour.

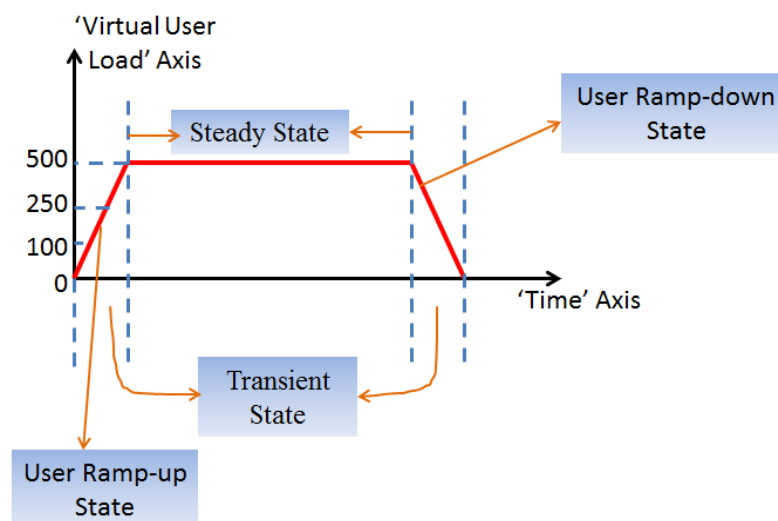


### Things to be noted :

1. Each virtual user request has to hit the web server just as an actual user in production.
2. The application is subjected to incremental levels of load tests. Meaning, Round 1 of test would be a single virtual user test to verify if transaction flow is in working condition. Round 2 would see a load of 5 simultaneous virtual users. Round 3 would be 20 simultaneous virtual users and so on till the final test of 500 virtual users. It is not advisable to test the application with maximum load at first GO itself, since this might lead to erroneous conditions in the environment causing test results to be skewed for any further tests. Also, the ramp-up load tests allow the application to warm-up till we reach final test round.

And if any performance degradation is observed at lower load levels, they can be looked into for resolution at that stage itself before increasing the load in next round of testing.

3. Note that while talking about virtual users, many use the terms “Concurrent” and “Simultaneous” interchangeably. However, to me *Concurrent users* are those who do the same action at the same time, however *Simultaneous users* are those who are active on the system at the same time but may not be doing the same action at the same time. I believe *simultaneous virtual users* represent the actual user behaviour in Production rather than *concurrent virtual users*.
4. Most load tests, irrespective of user count, are generally run for 1 or 2 hours to be able to collect necessary monitored statistics. This means that all virtual users need to be active on the system for steady state duration of 1 or 2 hours (refer Figure below).



*To be continued in next issue...*







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

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

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

If interested, connect on:

LinkedIn:

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

Twitter:

<http://twitter.com/SamarjeetM>



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

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

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

If you think that **YOU** are the **PLAYER** then send your profiles to [trainers@qualitylearning.in](mailto:trainers@qualitylearning.in).

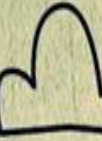
Click [here](#) to know more







are you one of those  
#smart testers who  
know d taste of #real  
testing magazine...?



then you must be telling your friends about ..



Tea-time with Testers

Don't you ? 😊



Tea-time with Testers !

first choice of every #smart tester !





# testing intelligence

- *its all about becoming an intelligent tester*



an exclusive series by **Joel Montvelisky**

## 10 reasons why You are NOT a Professional Tester! — Part 2

In last issue I published Part 1 of this series about why do I think testers are not treated professionally in some organizations.

My take is simple and I put the bulk of the blame on us The Testers, because many times we bring this upon ourselves by not taking our jobs seriously enough and not behaving professionally in our work.

It was nice to get some encouraging comments from testers I respect, but what I'm after is additional inputs on the subject. Even if you don't agree with me, I want to hear your feedback in order to learn from it and improve our work!



## A look back at the first 5 reasons

I will not go over each of the reasons I already talked about, but it is good to list them here for reference and completeness.

I believe that at least part of the reasons why You are NOT treated as a Professional Tester in your company are:

1. You think testing is not a technical profession, and so you don't even try to understand the code behind your product!
2. You are not involved in the process until you are hit in the head with a build by development and told to "go and test it".
3. Your only interaction with a Customer is when your Support Team asks you to reproduce a bug from the field.

4. Risk management is something you practice only in the context of Life Insurance.

and

5. You don't have a plan to improve the value of your testing.

Now a look at the next 5 reasons

Why You are NOT a Professional Tester!

## 6. You think your job is mainly about writing and running predefined Test Case Scenarios

There is **so much more** than only running scripted tests:

- Providing feedback on the design of your application.
- Analyzing the Risks of your current development plan and project.
- Providing informal feedback during the development stages.
- Developing an automation framework that will help your developers maintain the stability of the product while they work on it.
- Running tests, but definitely not only those you scripted before hand.
- Analyzing the results of your tests and the rest of the information available to you, to provide insights into the status of your product.
- Providing feedback on the process.

And I could go on & on...

In short, the value of your job goes way beyond executing test-steps and setting them to pass or fail!

## 7. Automation (and scripting) is an *Advanced Science*, and a project you will work in the future – in your spare time.

STOP coming up with excuses why not to work on automation!! This is another side of the technical shortcomings of some testers but from a different perspective.



Automation is not a magic pill or the cure to all the problems faced by testers, this is only a sales-pitch-lie from many tool vendors. But still, there are times when using scripts or tools to do part of your dirty-work will make it more efficient and save you time.

The problem is that, again here, *some testers* feel they are not technical enough to do this, and so they choose not to use automation or scripting to improve their testing. In a sense it is like striking stones or rubbing sticks to light a fire, and refusing to use a lighter while saying that for you it is easier this way...

## 8. You do most of your testing while standing high on top of you Ego

A good tester is a **humble tester**! We need to know how to provide feedback, and even more importantly how to receive feedback from teammates and peers.

Many testers get frustrated when team members (especially programmers) give them unrequested feedback on their testing, or when they are queried on a bug that was not found or a test that was not run. Many times there are good reasons for all these “misses” and we only need to keep calm and share this information, but lot’s of testers take these questions as personal attacks on their professional integrity and reply with loud tones or harsh words.

In the same way as you need to know how to report your bugs and provide negative feedback to your project team, you need to know how to receive constructive criticism from your peers.

No one expects you to be perfect, but they expect you to be professional about your mistakes and to learn from them as well as from the feedback you get from the team.

## 9. You don’t keep track of your professional skill set and the areas where you need to improve next

One of my best managers in the past used to talk about our personal “Virtual Toolbox” as the set of skills each of us carries with him and uses when needed.



- Do you know what tools you carry in your toolbox?
- What tools are in need of improvements or updating?
- Which are the tools that you always need, and that you may want to acquire next in order to improve the quality of your work?

Testing is without a doubt a craftsmanship, and without the proper tools (virtual and actual) you will not be able to create the required product.

## 10. The only idea you have about a career path involves becoming a manager or moving on to another career

Some people get into testing because they think it is a good path into programming. Others do because they don't know what testing is about and it sounds cool to "play" with applications all day long. After all, how hard can it be, right?

Part of them can end up been good testers (at least I hope that I did!). But most of them will end up frustrated, counting the days until they can stop testing and start doing the work they really wanted to do. While others don't appreciate the real challenges of testing, and think the only way to move forward is to start managing people.

It is true there are challenges and rewards to managing a testing team, but there are also countless disciplines to conquer that are not related to management and that may give you even more challenges and bigger rewards (and definitely a lot less headaches!)

My point is that, if all the time you are looking to do something else and not focusing on how to test better, there is no way you can do it more professionally. So think if you are in the right place, or if maybe you should simply be looking for something else...?

### Want to be professional? Start by looking at testing as a profession!

Looking at these ten points from 20,000 feet I think the line connecting them is the call to change our general approach to testing.

The first step is to start considering testing as **OUR Profession**.

Once we absorb this first step, the second one is to look at what we are missing in order to become better testers. What areas should we develop? How do we need to approach our work and the relationships with our customers and teammates? And what can we do NOW in order to increase the value of our work?

The third and last step (at least for this short approach) is to plan ahead how to improve, and to realize that as a profession we have much to learn before considering ourselves gurus or experts (if there is such a thing...)

The important thing is to realize that the change needs to come from within, and not from some God-given decree or from the title next to the name in our email's signatures.







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

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

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

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

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

[Back To Index](#)



# Teach-Testing →

An Ambitious Campaign by Tea-time with Testers




[Click here to Cast Your Vote](#)



by





# Call for Articles !

## Have you got something to say?

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

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

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

Submit your articles, stories, thoughts around software testing.

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

**Click [HERE](#) to read our Article Submission FAQs !**

# T ' Talks



*T. Ashok exclusively on software testing*

## Properties of a Good Test Scenario/Case

We all design scenarios/cases to test system. What are some properties that a good set of scenarios/cases should possess? This post outlines the properties that a good test scenario/case should satisfy.

A test scenario and associated test cases prime objective is to uncover potential issues in the entity under test. In the process we would like to ascertain correctness.

The properties outline this line of thinking.

1. A given scenario and associated test cases should be clear on what it is validating i.e. what entity it is testing. This is common parlance is understood as "**Requirements traceability**".
2. It should be clear as what type of defect it has the power to uncover. This in HBT (Hypothesis Based Testing) is called "**Fault traceability**". This makes the scenarios/cases purposeful.

3. That the number of scenarios/cases shall be proven to complete. This is a “controversial” statement. In HBT, this property is called as “**Countability**” i.e. that the number of scenarios/cases are no more or no less. This can be arrived only if the intended behavior is modeled and then scenarios generated. The number of scenarios can be arrived by combining the conditions logically and the corresponding test cases generated by combining the values of the data satisfied by the conditions.
4. That the test scenarios/cases should be staged into quality levels rather than being one to uncover a variety of defects so that these are small and purposeful.
5. That in addition to staging by quality levels, it is good to group scenarios/cases by types of tests to enable better clarity and purposefulness.
6. The number of scenarios/cases as we progress from the lowest quality level to the highest is shaped like a frustum of a pyramid. i.e. number of test scenarios/cases are lowest level of quality is indeed a lot compared to the numbers at higher levels.
7. That the distribution of positive and negative test scenarios/cases is indeed good enough. Hmmm – how do we know this? Extending point (3), negative scenarios are generated when conditions that are violated are combined while negative test cases are generated when values of data that do not satisfy the conditions are combined. Given this more conditions implies more scenarios (and also negative scenarios) while more data values implies more test cases. As one proceeds from lowest to highest quality level the distribution of -ve:-+ve skews on lower side. i.e. at higher levels, the scenarios/cases are more conformance oriented.
8. That the scenarios/cases are ranked in terms of priority guided the entities that they test and the types of defects that they defect, to enable intelligent choosing of which to execute when faced with crunch time.
9. To aid in ensuring scenarios when automated run as unattended and as long as they, it would be useful to design the execution order of scenarios. i.e which scenario to execute in case the current one fails. This can be factored into automated to allow for “long runs” intelligently.

Understanding properties can enable us to assess the efficacy of scenarios/cases and also yield higher efficiencies. In HBT this is captured in the HBT test case architecture whether nine properties allow segregation of scenarios/cases in a beautiful onion peel shell. The form and structure of this architecture enables better clarity and thus improves effectiveness & efficiency.

Have a great day!



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

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

He can be reached at [ash@stagsoftware.com](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.



# Testing PUZZLES

by Sebi



Claim your **Smart Tester of The Month** Award. Send us an answer for the Puzzle and Crossword bellow b4 15<sup>th</sup> May 2012 & grab your Title.

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

**Exciting  
PRIZE for 1<sup>st</sup>  
three  
WINNERS\***

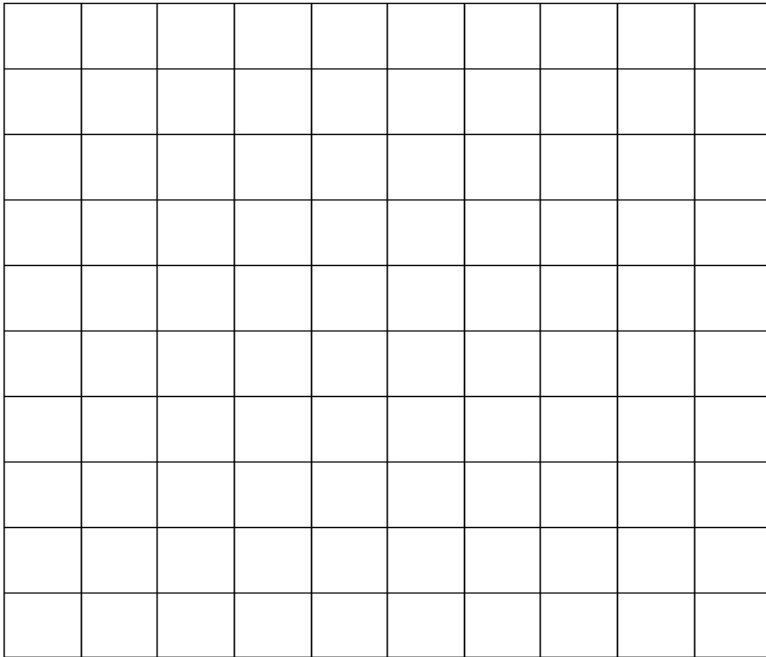
**NOTE :** S.T.O.M. contest comprises of Testing Puzzle + Crossword. To claim their prize, participants should to send answers both for puzzle and crossword.

# Puzzle “Help that man”

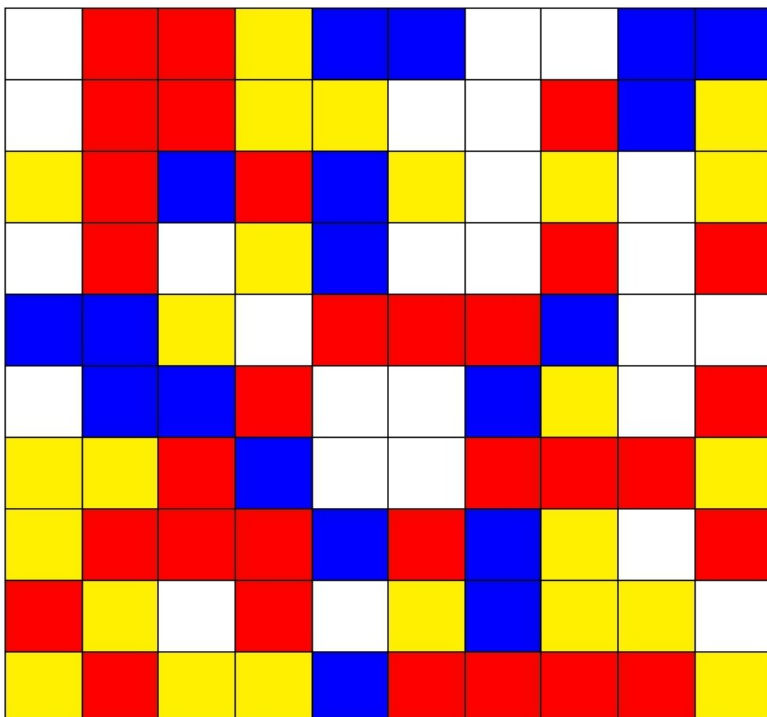
"A man is walking on the board starting at an unknown position. The board is a 10x10 matrix (see fig 1). He can only go in these directions:UP,DOWN,RIGHT,LEFT. When he steps on a white cell it becomes red, when he steps on a red cell it becomes yellow, when he steps on a yellow cell it becomes blue and when he steps on a blue cell it becomes white again.

This puzzle consists in presenting a solution for a possible route, starting from any position you want, with as many steps needed, to color the board as in fig2.jpg.

Please give the coordinate paths (example (2,3) (2,4) (3,4) ) or an animated solution, or starting position followed by the steps UP,UP,UP,DOWN,LEFT etc..)



Picture 1



## 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.

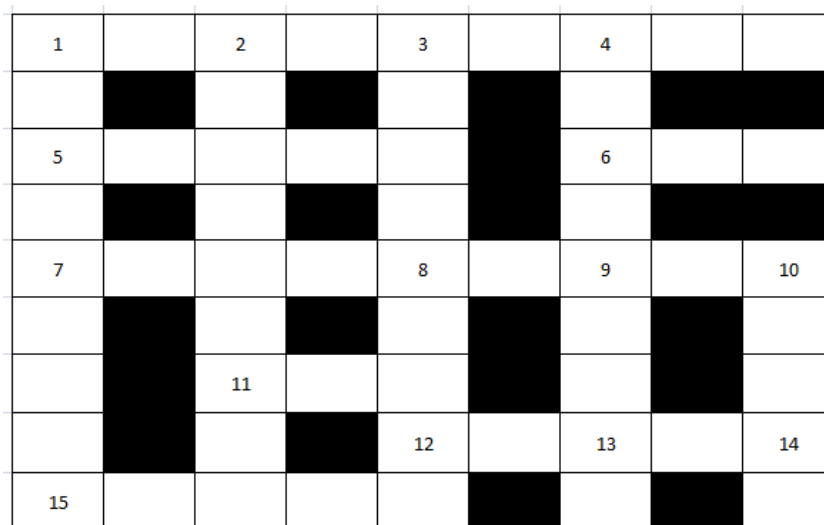
You can find some interactive testing puzzles on his website [www.testalways.com](http://www.testalways.com)

Picture 2





# TESTING CROSSWORD



## Horizontal:

1. He is one of the uTest's Board of Director (9)
5. The first word of "Multi Unit Testing" (5)
6. A fault in a program which causes the program to perform in an unintended or unanticipated manner. It is called \_\_\_\_\_? (3)
7. Jason Huggins, CTO belongs to which company, first word? (5)
8. The first executable statement within a component, in short form (2)
9. A computer system to analyze, understand and generate natural human languages. It is called \_\_\_\_\_ (in short form) (3)
11. It is a Microsoft's web browser, in short form (2)
12. It is an open source test suite for automated conformance and functional testing of various Linux distributions against LSB standard requirements on base system interfaces behavior (5)
15. It is the main markup language for web pages (4)

## Vertical:

1. Author of "Secrets of Buccaneer - Scholar" book (9)
2. The \_\_\_\_\_ Automated Testing Framework is a UI module-based automated testing framework for web applications (9)
3. It is a continuous activity which goes hand in hand with development, is called \_\_\_\_\_ testing (5)
4. It is a unit testing tool for programmers and testers developing software in C or C++ (7)
8. It is a discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition (5)
10. Two testers work together to find defects, is called \_\_\_\_\_ Testing (4)
13. A version number; version date, or version date and time stamp, is called \_\_\_\_\_, in short form (2)
14. Continuously raising an input signal until the system breaks down. It is called \_\_\_\_\_, in short form (2)



# Answers for last month's Crossword:

T	O	S	C	A	P	A	C	H	E
A		O		X		C			N
R	I	A	T	E	S	T	U	B	D
A		K				U			U
N	T	I	M	E	U	A	T	D/H	R
T		E		I		L			A
U	T	E	S	T	R	A	C	C	N
L		E		G			M		C
A	D	H	O	C	S	V	M		E

Answer for last Puzzle:

The Expression Fails only for prime number when all the fields contain prime numbers than the result is displayed as 100.



*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

### My Voice on "Teach-testing"

I support every bit of this campaign. Glad to see this initiative. Please take the steps to conduct such programmes in colleges because I read that there is a massive lack of professional qualities and technical knowledge in students which are needed for a engineering profession.

Your this initiative will surely bring the change.

- Arthi Vivek

Dear Editor,

I liked your editorial of last month and I second your thoughts.

All articles that you publish are really good and can make significant difference if we readers start implementing those ideas, tips and thoughts in our testing practices.

Thank you.


- Ritika Ahlawat

Good to see an Indian magazine crossing 90 countries. Hip hip hurray !

- Kiran Pathak

# 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 -



IT'S  
ALWAYS  
TEA-TIME

Jerry Weinberg

T Ashok

Joel Montvelisky

Mike Talks

Anurag Khode

Bernice Ruhland

Samarjeet Mohanti

# our family

## Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

## Contribution and Guidance:

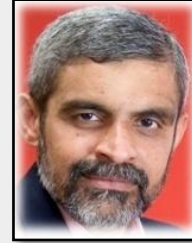
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

## Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover Page Image- Roses and Teacups

## Core Team:

Anurag Khode (Nagpur, India)

Dr.Meeta Prakash (Bangalore, India)



Anurag



Dr. Meeta Prakash

## Testing Puzzle & Online Collaboration:

Eusebiu Blindu (Brno , Czech Republic)

Shweta Daiv (Mumbai, India)



Eusebiu



Shweta

## Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)



Kiran Kumar



Chris



Romil

*// Karmanye vadhikaraste ma phaleshu kadachna /  
Karmaphalehtur bhurma te sangostvakarmani //*



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



Join our community on



Follow us on



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

