THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

MARCH 2012 | YEAR 2 ISSUE II

TEA-TIME WITH TESTERS

First Indian Testing Magazine to reach 89 Countries in the world !

Created and Published by:

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

Editorial and Advertising Enquiries:

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

Editorial

Are you a PRACTITIONER or just a READER ?

Couple of days before, I met one lady in my office cafeteria. I did not know her personally but since she was sitting with my other tester friends we got a chance to converse.

They all were discussing about testing conferences, magazines, discussion forums and their active participation in it. As topic turned to contribution and participation, I saw that lady getting little nervous. I asked her what the issue was. She said that she wants to participate in conferences, wants to be part of different discussions on testing forums but she runs short of information, references to support her views.

I asked her if she reads any testing magazine or follows any blog. She said that she has *subscribed* to almost all testing magazines, different blogs but still nothing is working out for her. I asked her if she ever tried to practice, implement what she learnt by reading those magazines and blogs. Lady got angry and argued that since she has *subscribed* to many magazines & forums, she expects herself to become knowledgeable tester. She wanted to be an expert just by *reading* things.

I smiled and left the discussion.

Well, do you see the problem here? Do you also believe that having just *subscription* in place or reading things superficially is going to make you good tester? Of course NOT ! What you need to be is a GOOD *reader* first. By GOOD reader, I mean the one who practices what he reads.

If you ask me, whatever craft of testing I have learned so far is by *practicing* things that I got to learn by reading. Unless we practice, we won't have experience and we won't have doubts too. Without an experience or at least a doubt, I don't think one can ever participate in discussions.

If you want to earn benefit from what you read, you got to practice it. I request all young TTWT readers to practice things that you are getting to read through this magazine. Mugging up those definitions might give you certificates but you can not be great at testing until you practice its craft.

We are trying our best to offer you real and practical knowledge. Practice it once and see the difference.

Enjoy Reading !

Yours Sincerely,

Bhamare

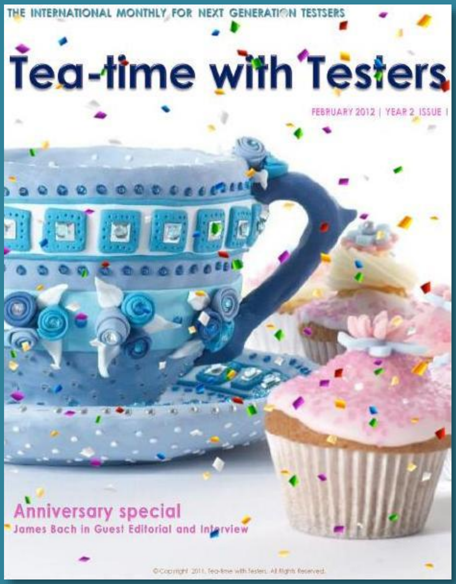
- **Lalitkumar Bhamare**

editor@teatimewithtesters.com

QuickLook

Testing Puzzles
by Sebi

Crossword by


THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTSERS

Tea-time with Testers

FEBRUARY 2012 | YEAR 2 ISSUE I

Anniversary special

James Bach in Guest Editorial and Interview

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

February 2012

What kind of tea you drink to inspire your creativity ?

James Bach's editorial is superb, as we've come to expect from James. I believe the idea that "testing is dead" is not a belief, but a hope or wish pretending to be a belief. I think we all can agree that it would be wonderful if we could get quality software without testing, but it would also be wonderful if I won the lottery or magically cured our children so they were never sick. The fact that these things would be wonderful does not mean they will happen. Not in our lifetimes. Not ever. Testing, like death and taxes, will always be with us. If we don't do it during our development process, our clients will surely do it when they try to use our products--at which point the products may be found wanting.

I enjoyed the systematic way Mike Talks covered the subject of defects. The article should prove extremely helpful to testers and their managers. I do think, however, that the article would be improved if he said something about "cost of repair" as a component of "severity." A defect could have important consequences but require a total rewrite of the system to correct it--which may be deemed impractical. Perhaps the reason people don't mention "cost to repair" is that the testers may not be in a position to evaluate that cost.

On the other hand, "consequences to our organization" may also be impossible for testers to evaluate. I would have liked to read Mike's ideas on whose job it is to write defect reports--the people involved and the process they should use. Maybe he'll do that in a future article.

Anne-Marie Charrett's coaching protocol is a super addition to Tea-Time with Testers. Everybody could benefit from this kind of meta-coaching. I hope Anne-Marie gives us more such protocols in the future.

Bernice Niel Ruhland's article on Journal Clubs is another type of article we need to see regularly, both about varieties of "Journal Clubs" and also about running effective meetings of all the many kinds in which testers participate. It's important not to convey the impression that there's only one right way to run such meetings.

I think many of my clients would greatly benefit from testing what they call their "Quality Agenda" against Joel Montvelisky's SMART criteria. I'm afraid that most of them would fail this test, but like all good testing, it could teach them things of importance.

T. Ashok's metaphor of drug potency turns my mind to a number of interesting situations. Unfortunately, I'm not sufficiently familiar with medicine to make full use of this rich metaphor. I hope that's not true of all testers.

In sum, another wonderful and useful issue.

I don't know what kind of tea you drink to inspire your creativity, but keep it up.

- Jerry Weinberg

Hi,

I liked T Ashok's 'Test Case Potency' article very much.

I have decided to take its printout and circulate it to my team members on occasion of my Birth day !

- Kiran Kumar

Wind River Android Testing Software Wins Embedded AWARD 2012

ALAMEDA, Calif., Mar 14, 2012 (BUSINESS WIRE) -- Wind River, a world leader in embedded and mobile software, is a winner of the embedded AWARD 2012 in the Tools category for its Wind River User Experience (UX) Development Kit, a kit that enables the rapid creation of automated test scripts for Android devices.

Earlier this month at Embedded World 2012 in Germany, an independent jury of experts selected outstanding technical innovations in embedded technologies in two categories: Hardware and Tools. With only three winners awarded for 2012, these highly selective awards pay tribute to exceptionally innovative products or developments that are unique and forward thinking. Drawing more than 800 international exhibitors, the embedded world exhibition and conference is among the world's largest events for the embedded industry.

"As Android devices continue to flood the market, the industry is recognizing that poorly tested applications can result in significant costs if the user experience is unsatisfactory. In order to deliver a quality device that a consumer will want to purchase and use, thorough automated testing is needed," said Amit Ronen, vice president and general manager of device test at Wind River. "The embedded AWARD is a great achievement that acknowledges Wind River's efforts to help customers develop Android devices with reliable and fulfilling user experiences in order to create greater customer satisfaction to improve the bottom line."

Wind River UX Test Development Kit is designed to assist in the validation of the user experience of a device by reproducing human interactions to test user interfaces. For example, a quality assurance engineer can automate scenarios that test the experience of interacting with graphic interface elements such as dragging items or typing entries on a physical or virtual keyboard. The fully automated execution of these tests can replace a significant amount of manual testing and dramatically reduce testing cycles and time-to-market. The test scripts are managed and automatically executed using Wind River Framework for Automated Software Testing (FAST), an automated software testing framework for Android-based devices.

A video demonstration of Wind River UX Test Development Kit can be viewed at http://www.youtube.com/watch?v=NhXMiit4Abo. To learn more about Wind River Android software, visit http://www.windriver.com/products/mobile/.

Credit: Business Wire

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

Back To Index

Announcing...

Smart Tester of the Month Awards !!!

❖ Winners for Testing Crossword :

Name: Sonal Agarwal
Designation: Software Engineer
Company Name: LogicNEXT Softwares & Services Ltd
Place: Noida

Name: Saritha S.P
Designation: Lead Test Analyst
Company: Hitech Outsourcing Services
Place: Kochi

Name: Priya Verma
Designation: Sr. Test Engineer
Place: Mumbai

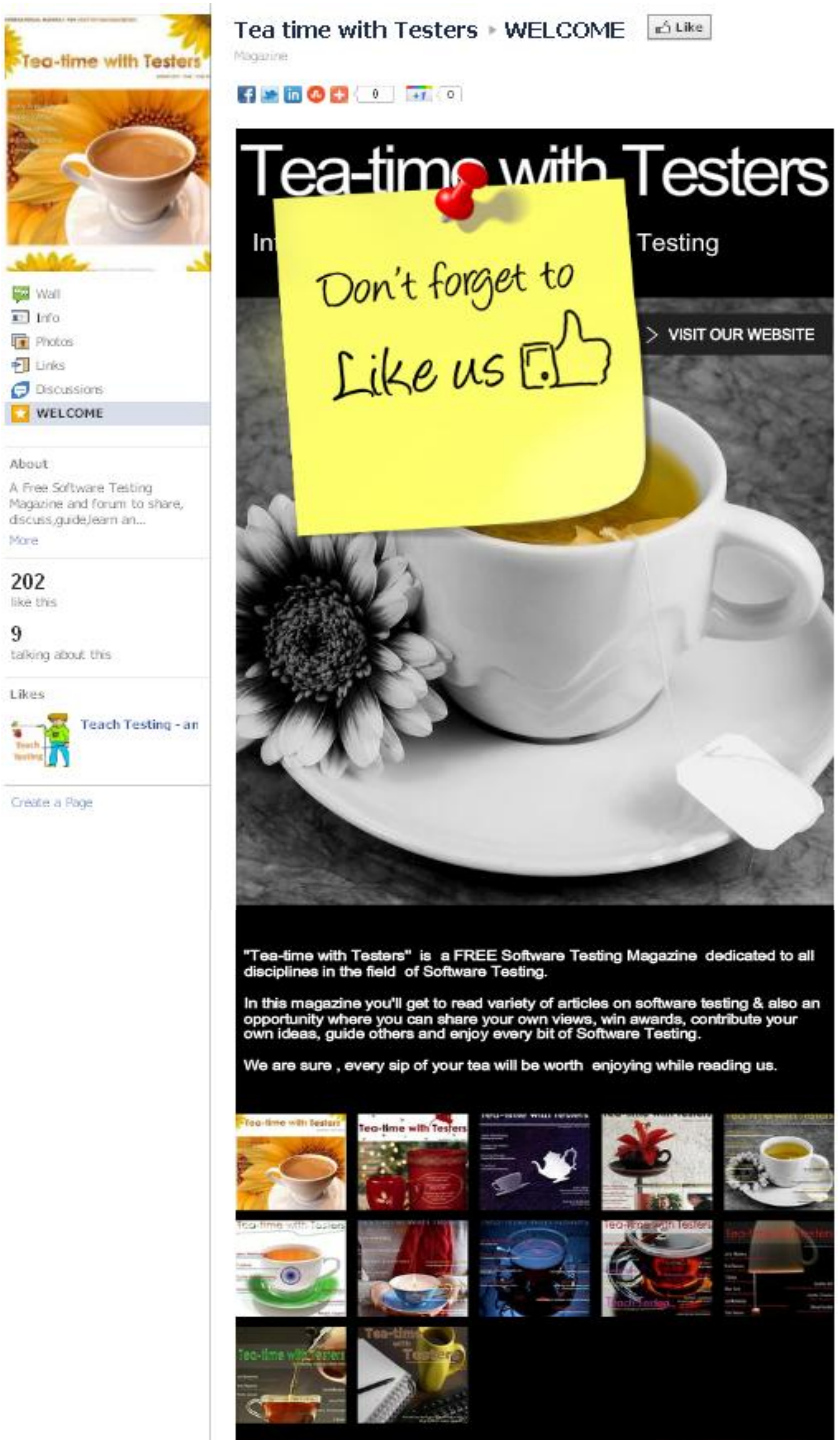
Congratulations !

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



Tea time with Testers ▸ WELCOME

Magazine

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" 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

Fourteen Months

10000+ Readers

89 Countries

ONE Magazine

TEA-TIME WITH TESTERS

Subscribe here Right Away to get our all Issues for FREE

Why Congruence is Essential for Managing (Part 1)

You're writing a gospel

A chapter a day,

By things that you do

By things that you say.

People read what you write

Whether faithless or true,

Say, what is the gospel

According to you?

- Children's poem

Image by : kinoanoa

To produce high-quality software, we need high-quality, effective software engineering managers. This article is about how to become such a manager—such a person.

Up until now, such managers have been rare in software engineering. During my career as a software developer, I never had such a manager, so I believed they didn't exist. More than that, I believed they were unnecessary—and I was right. Such high-quality managers *were* not necessary to produce mediocre software. In my years as a consultant, however, I've met many high-quality managers, and learned they are necessary if you want to go beyond mediocrity.

Perhaps your experience is much like mine. Perhaps you haven't met many high-quality software engineering managers. In such a case, you'll have to take my word for their necessity, because if you do not believe high-quality software requires high-quality managers, this book seem irrelevant to you.

But even if you believe in the necessity of effective managers, you may not agree with my understanding of what it takes to be effective. For instance, you may believe an effective manager simply needs to *understand* the concepts of software engineering, not to *practice* according to those concepts. This erroneous belief is the cause of a great deal of trouble, and extremely difficult for intelligent people to overcome. At least it was for me. For many years, I enjoyed the superiority of thinking that if I *knew* what to do, it didn't much matter if I actually did it. That opinion made it easy for me to be a sideline manager.

I've met many others like me in the software business, so this first chapter will introduce the idea of congruence, and tackle the question: Why must actions be congruent with thoughts?

1.1 Knowing versus Doing

In order to manage an engineering system by feedback control (Figure 1-1) a manager as controller needs to

- plan what should happen
- observe what significant things are really happening
- compare the observed with the planned
- take actions needed to bring the actual closer to the planned (Figure 1-2).

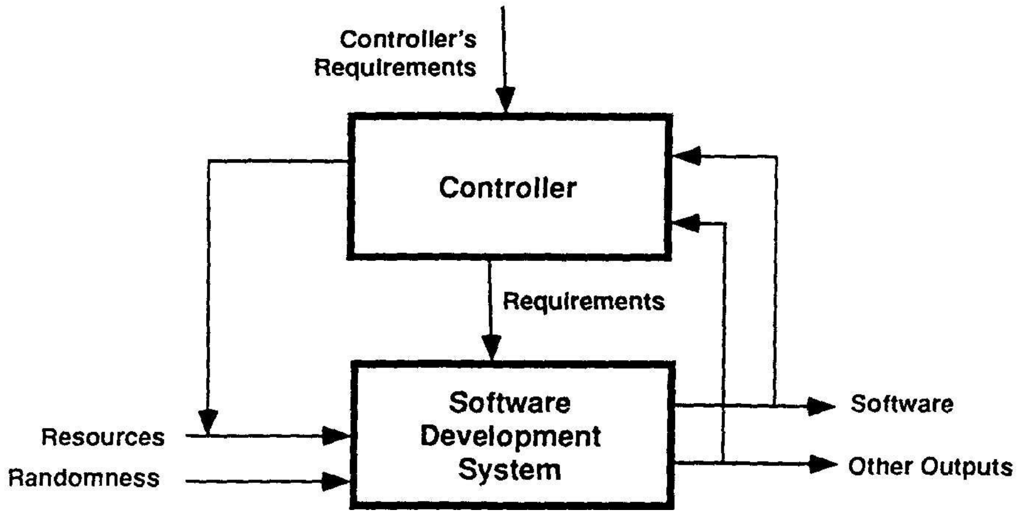


Figure 1-1. The feedback model of a software development system requires feedback of information about the system's performance plus requirements for the controller to compare with that information. This is the model that distinguishes Pattern 3 (Steering) from Pattern 0 (Oblivious), Pattern 1 (Variable), and Pattern 2 (Routine).

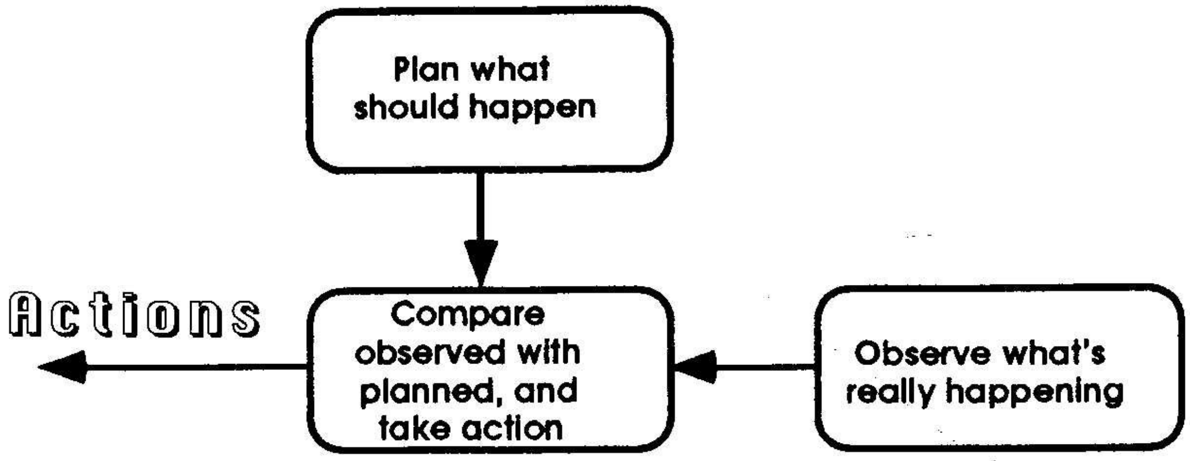

Plan what should happen
Actions
Compare observed with planned, and take action
Observe what's really happening

Figure 1-2. In Pattern 3 (Steering) organizations, management's job is to control a process that produces a desired product or service. Management plans what should happen, then observes what actually happens. Management's actions are designed on the basis of the difference between the planned and the actual results, then are fed back into the process being controlled.

The focus of this article is on the actions.

The focus on action may seem strange to those readers who believe that once the planning is done, the right actions must follow. I used to believe this myself. I had experienced many ineffective managers, and I attributed their ineffectiveness to their lack of brains: they simply didn't know what to do. Much later, I learned this wasn't always their problem. I have witnessed hundreds of instances when managers knew quite well what to do but were somehow incapable of doing it.

For instance,

• A manager knew there was not the slightest hope of completing a project on time, but was unable to say this to her manager in order to open a discussion of alternative.

• A manager knew adding developers to his late project would only make it later, but he was unable to take the risk of appearing to do nothing.

• A manager knew screaming at people would only make things worse, but was unable to stop screaming at them.

• A manager knew an employee smelled so bad the other employees weren't able to work with him, but wasn't able to speak to the employee about it.

• A manager knew a certain person was right for an assignment, but put another person there because he disliked the best candidate.

• A manager knew a project shouldn't proceed without a clear understanding of the problem they were trying to solve, but was unable to resist the eagerness of the developers to get started writing code.

• A manager knew she shouldn't make promises she wasn't sure she could keep, but she kept making promises to get out of difficult interpersonal situations (temporarily).

1.2 Requisite Variety

Ross Ashby pointed out that any controller, in order to be effective, must have sufficient variety in its coping mechanisms to counter the variety of actions that could be exhibited by the system being controlled. Ashby's *Law of Requisite Variety* says the action taken by the controller must be *congruent* with the situation, in there is at least one controller action to deal with each possible system action. The managers who couldn't do what they knew they should do lacked the requisite variety of actions to perform their jobs effectively.

For control purposes, it doesn't matter *why* these managers were unable to exhibit the requisite variety of action. A manager (acting as controller) may lack a plan or may lack the observations needed to compare the plan to what's really happening, as Volumes 1 and 2 discuss extensively.

But it's also useless if a manager knows what to do but is incapable of actually doing it—especially under stress, when managers may be called upon to use any of their full variety of potential actions.

When people are not tapping their full variety of potential actions, they are coping *incongruently*. According to the Law of Requisite Variety, managers acting incongruently may not be capable of controlling the system they are trying to control. This book is about increasing the congruence of your actions—especially under stress—and thus your ability to manage for the quality you desire.

1.3 The Importance of Management

The significance of congruent management extends beyond simple projects to the process of raising the general cultural level of the organization. As Bill Curtis—former director of the Process Program at the Software Engineering Institute (SEI)—observed:

If an organization is trying to build a general management infrastructure at the same time it is trying to improve its software process, then it will take longer to achieve the next level. Even further, Humphrey and Curtis—SEI's first two directors of the Process Program—said:

A defined engineering process cannot overcome the instability created by the absence of sound management practices. Occasionally, unusually capable and forceful managers can withstand such pressures...

Now, it seems to me we could follow this observation in one of two directions.

1. We could try to introduce defined engineering practices so we could overcome instability even with incapable and weak managers.

2. We could seek and cultivate these unusually capable and forceful managers. The SEI is pursuing Path 1, which is appropriate for an organization such as the SEI whose clients are organizations. This volume will pursue Path 2, which is appropriate because that's my audience—individuals who want to become "unusually capable and forceful managers."

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 .

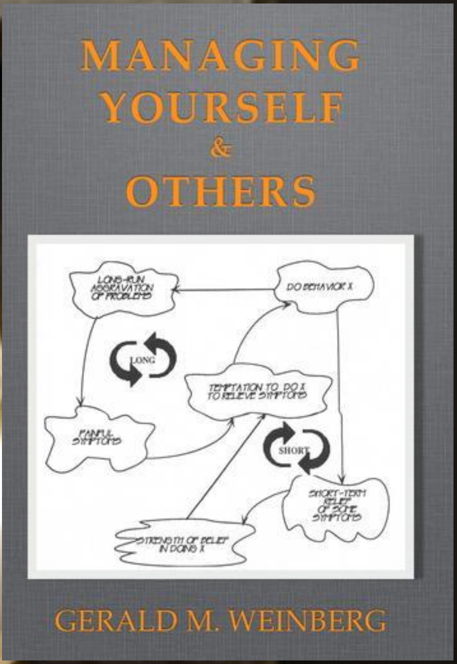
Gerald can be reached at hardpretzel@earthlink.net or on twitter @JerryWeinberg

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
LONG-RUN AGGRAVATION OF PROBLEMS
DO BEHAVIOR X
LONG
TEMPTATION TO DO X TO RELIEVE SYMPTOMS
PAINFUL SYMPTOMS
SHORT
SHORT-TERM RELIEF OF SOME SYMPTOMS
STRENGTH OF BELIEF IN DOING X
GERALD M. WEINBERG

TTWT Rating: ★★★★★

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

5 MYTHS... BUSTED!

About Testing Regulated Software

By John McConda

"We can't do that, we'll never pass the audit!"

If you've worked in regulated software, chances are you've heard that statement. The threat of an audit can cause even the most enthusiastic proponents of new ideas to cower in fear of the dreaded "finding," an auditor's report on something noncompliant in your process. The truth is, many of the most cited "can'ts" about regulated software testing are nothing more than myths.

Myth 1: We Can Only Test with Prewritten Test Cases

Scripted manual tests have been used for so long that many organizations assume that they are the only option, but that doesn't mean this is the only acceptable approach. When someone says, "We can't do exploratory testing," what he is usually afraid of is not being able to present objective evidence of testing, such as the kind called for in FDA regulatory guidelines [1]. Those regulations require that someone looking at your documentation can tell what you tested without taking your word for it. There is nothing about pre-scripted tests that make them better for this than exploratory tests. Some problems with test evidence, like putting a checkmark in the Pass box or just typing "as expected" for a result, are actually unique to pre-scripted test cases. In this regard, exploratory testing can be more suited to showing objective evidence because notes on observation make up the bulk of the documentation. An exploratory test session is designed so that the tester has objective evidence about what he saw.

Screen and video capture tools can make gathering objective evidence even easier. One tool, SiriusQA's Test Explorer, is specifically designed to support the recording of objective evidence during exploratory testing [2]. Other tools I've used that feature video capture are BlueBerry Test Assistant and HP Quality Center.

Myth 2: Test Automation Is Too Difficult

Test automation is a tricky road to walk in any context, but in the regulated world, misunderstood rules often remove it from consideration completely. This is unfortunate because automation can supplement a regulated testing process just as much or more than a non-regulated one. An automated test can add control to your evidence because it logs results the same way every time, but some auditors may ask you to validate each tool for your specific process to make sure it meets their standard of objective evidence. Types of tool validation that I have seen are *general* and *specific*. General validation is usually a certification for a certain industry. Some companies will tout this certification about a particular tool if it has passed an audit by a regulatory body in that industry. Second, the tool may need to be validated for your specific environment. In my experience, our team had to prove that each automated test's results were identical to the same test run manually. If your regulations require a high level of validation before use, you may have a difficult choice between the amount of time automation will save you in the long run against the additional cost you will incur for tool and test validation. On the other hand, if you are spending a substantial amount of time running a large suite of regression tests manually, you may still come out ahead.

Automation is more than just regression testing, though. It can also be used to run high volumes of test data through thousands of combinations that would take months to do manually. In his article for StickyMinds.com, "The ROI of Test Automation," [3] Mike Kelly lists even more types of test automation, many of which, if done in addition to a validated manual test process, may not even need to be a part of your officially validated process (see Myth 5).

Myth 3: Regulated and Agile Don't Mix

Over the past ten years, agile software development has taken the world by storm, but regulated software has been one of the last holdouts. Some of this can be attributed to one of the Agile Manifesto's stated core values of "Working software over comprehensive documentation [4]." As agile processes have evolved, so have ideas about where they can be used. Today, FDA-regulated companies like Abbott Labs are presenting case studies that describe how agile processes have been used in developing their applications [5], and Department of Defense-regulated companies like Lockheed Martin are also publishing case studies of their results with agile [6]. The common thread among many of these case studies is that software teams were able to cut through perceived notions and get to the real purposes and constraints of the regulation. Once those were understood, then it was possible to tailor agile processes to the regulatory requirements.

Rich Sheridan, CEO of Menlo Innovations, a biotech software firm, believes agile processes are essential to software that is developed for medical devices. He talks about many agile ideas his teams implement like "weekly show and tells where the customer runs the software in front of our team," automated unit testing, and paired programming. He adds: "In our environment, no production code can be produced by anything other than a pair of developers, thus each line of code is reviewed by two sets of eyes. Regardless of what the FDA may think about such a system, I know I wouldn't want to have a device hooked to me that wasn't developed in a test-driven, paired-programming environment [7]."

Myth 4: The Auditor Is Your Enemy

When most of us hear the word auditor, we think of a grumpy IRS agent with horn-rimmed glasses and a thin black tie. In my experience, the auditors I've encountered were quite willing to work with our team to make the process as smooth as possible. The most effective way to avoid significant findings is to know what you're being judged on as you conduct your process and to make sure everyone on the team does, too. At one of my former companies, we put together an agenda for the auditor to approve and then put all of our evidence into binders in the conference room so she could peruse at her convenience. We made every effort to be transparent, so she had no reason to suspect we were covering up a flaw in our process. If we did have a known flaw, we would document it for corrective action up front. Practitioner and author Matt Heusser, who has tested systems subject to HIPPA and SAS 70 audits, believes the perception of an adversarial relationship with auditors is counterproductive. "There is the perception that you never volunteer information to an auditor, so we were taught to answer questions and say nothing more. There was even a joke that, if an auditor asks if you know what time it is, the correct answer is "Yes." But even the word "audit" itself, comes from the Latin word for hearing. An audit should be a software team's invitation to "Come hear what we do [8]."

Myth 5: Regulated Testing Is Boring

A few years ago, when I was working on my master's degree, I talked after class one evening to a few fellow students who mentioned they were testers. Excited about having classmates in the same field as mine, I asked them how they liked their jobs. All three of them agreed, "I'm getting this degree so I can get out of testing. It's so boring!"

Bored often means you're not using your brain. Regulated testing is subjected to this stigma even more often, mostly due to Myth 1. Doing nothing but exe-cuting the same manual scripts over and over is enough to make any tester consider taking apart his cubicle with a screwdriver. Despite the ideas I described in Myth 1, your company may not be ready to move away from test scripts for their officially validated testing. If that's your situation, then there's always what I like to call Black Ops testing. The first time I brought up the idea of using exploratory testing for our regulated software, I encountered some resistance to making it part of our standard operating procedures. What we agreed to was a clause in our documentation stating that we "reserve the right to do other forms of testing as needed." This gave our testing team carte blanche to do as much additional testing as we needed to feel good about our coverage, without the burden of test cases. To succeed at this, though, you need to have time for your extra testing built into the project plan. If your project manager balks at this, make a case for just a few extra hours the first time. If you can show how much more efficient your testing is becoming, you'll have leverage to get the time you need.

Certain software is regulated for a reason. These applications are often life critical or they process sensitive information. Stakeholders deserve the best testing possible, performed by the best testers around. Unfortunately, the myths around regulations can keep good testers away and handcuff those who do test them. Regulations are intended to provide accountability, but it is up to those of us who create and maintain these applications to make ourselves accountable for testing with the best tools and techniques available—and not to be intimidated by myths.

References

1. http://www.fda.gov/downloads/RegulatoryInformation/Guidances/ucm126955.pdf: Page 6, Section 3.12
2. http://www.testexplorer.com/te_details.shtml
3. http://agilemanifesto.org/
4. http://www.klocwork.com/blog/?p=355
5. http://www.agilejournal.com/articles/columns/case-studies/313-case-study-war-stories-fighter-jets-and-agile-development-at-lockheed-martin
6. Rich Sheridan Interview via email: Conducted 11/23/09
7. Matt Heusser phone interview: Conducted 11/25/09
8. "The ROI of Test Automation" http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=8502

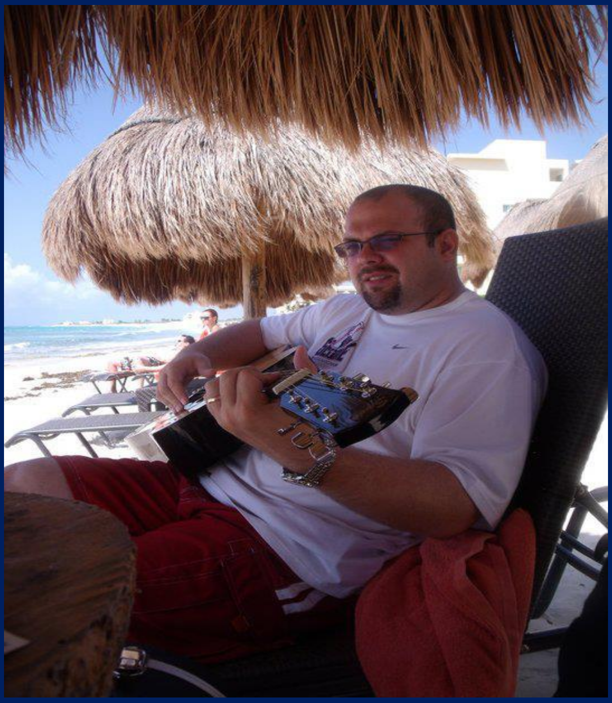
John McConda

A career tester, John McConda has been seeking to improve his skills and his teams' effectiveness for more than ten years.

In his current role, John consults for a high-profile US Government contract, providing training and strategy as well as getting his hands dirty with exploratory testing, automation, and performance. He is an instructor for the Association for Software Testing and co-founder of WREST.

He's also passionate about songwriting and being a dad.

Contact John atjohn@mcconda.com or @mcconda.



Back To Index

Image courtesy : MMVI New Line Production

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

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

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

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

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

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

~ PLEASE NOTE OUR NEW CONTACT DETAILS ~

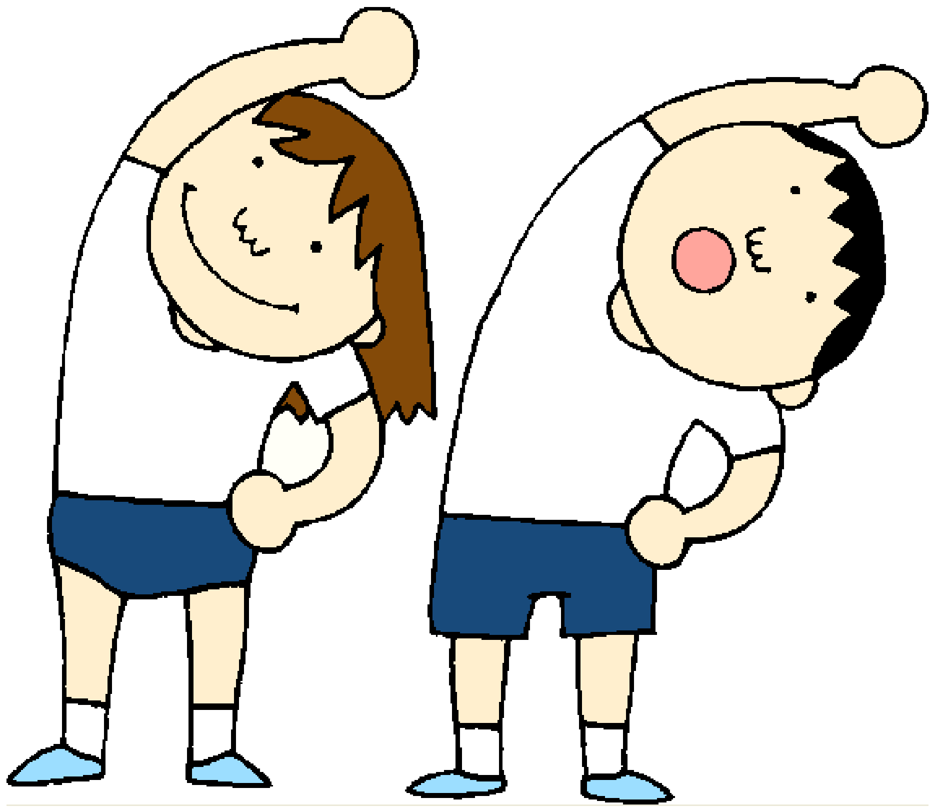
For Editorial enquiries, write to us on editor@teatimewithtesters.com

To contact us, mail us at 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 1



The focus of this two-part article is guidelines for training someone who is new to software testing. Part 1 discusses different sources of training. Part 2 focus is 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!

How much training is required?

Before identifying how to train an employee new to testing, you need to understand his understanding of testing concepts. Not necessarily formal software testing and knowledge of testing terms. Instead does he have any programming experience where he tested his own code? If yes, how did he test, what was his thought process? Does he have any hobbies such as building spreadsheets or databases to track information? Another avenue to pursue is how he approaches problem solving. Try to understand his level of testing and problem-solving experience as that will help you define his training program.

Training material

There are wonderful resources available through published books, blogs, websites, and articles. Below are a few free and paid training opportunities. To expand this list, connect with other testers to share articles and other avenues for learning.

Identify initial material to provide the tester with an overview of software testing and relationships. The book "Perfect Software and other illusions about testing" by Gerald M. Weinberg is a great starting point. Continue to introduce new material to expand testing concepts and ideas. As a rule of thumb, review the material for general understanding not memorization of terms. Some material should be re-reviewed at a later date to reinforce what was learned and to identify new information.

Do not forget to share this material with your other testers. No matter how much testing experience you have, there is always something new that can be learned.

Books:

- Perfect Software and other illusions about testing by Gerald M. Weinberg. This is an excellent book that the tester should read in its entirety. Then distribute this book to the rest of the department.
- Testing Computer Software by Cem Kaner, Jack L. Falk, and Hung Q. Nguyen. The authors recommend specific chapters for the new and more experienced testers making this a great investment for your department.
- Crucial Conversations Tools for Talking When Stakes Are High by Kerry Patterson, Joseph Grenny, Ron McMillan, and Al Switzler. This is not a software testing book. However, communication skills are critical for a software tester to develop. Sign up for their free newsletter to learn how to handle difficult conversations.
- Lessons Learned in Software Testing by Cem Kaner, James Bach, and Bret Pettichord. This book is excellent for both new and experienced testers.

Start with the following chapters:

  - Chapter 1: The Role of the Tester
  - Chapter 2: Thinking Like a Tester
  - Chapter 3: Testing Techniques
  - Chapter 4: Bug Advocacy
  - Chapter 7: Interacting with Programmers

- Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design by James A. Whittaker. This is a great book once the tester has more experience and is looking to expand testing approaches.

Free online courses:

- BBST Testing Course http://www.testingeducation.org/BBST/. This black box testing course is divided into different modules providing course slides and videos in manageable segments.

Paid courses:

- Rapid Software Testing by James Bach and Michael Bolton.
  - Refer to James's website for course outline:

    http://www.satisfice.com/info_rst.shtml
  - Refer to Michael's website for course outline:

    http://www.developsense.com/courses.html

Webinars:

- Open Lecture by James Bach on Software Testing. This is an excellent lecture that the tester should watch more than once as it is packed with valuable information. http://www.softwaretestingclub.com/video/open-lecture-by-james-bach-on-software-testing

Articles:

- **Tea-time with Testers** – have the new tester subscribe to this e-zine to receive emails for new issues. The following articles can be found in Tea-time with Testers.
- Testing Intelligence - its all about becoming an intelligent tester, by Joel Montvelisky. May 2011
- 'Investigating Bugs: A Test Skills Study', by James Bach. October 2011

- ‘Professional Growth: Is it unrealistic to do it on your own time’ by Lisa Crispin. December 2011

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.
- Share new learning material with your more experienced testers as they can always learn something new.

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.
- 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

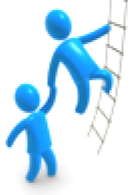
Training an employee who is new to software testing is different than an experienced tester. Prior to developing a training program, understand his background in problem-solving and testing. Many people have informal testing experience relating to a hobby or job responsibility not focused on software testing. Gradually introduce different training approaches through reading material and webinars.

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



Back To Index

Software Performance Engineering :

‘Tailored into SDLC – Do it Right’

by Samajeet Mohanty

In my first artefact of this series, we familiarized to the concept of Software Performance Engineering and the possible pit-falls of not following it. This article will delve into the details of where exactly does it fit into the service/product life cycle.

As we know by now, ***Software Performance Engineering*** ***(*** *referred as* ***SPE*** *in rest of article****),*** *is a software-oriented approach and focuses on architecture, design and implementation choice.*

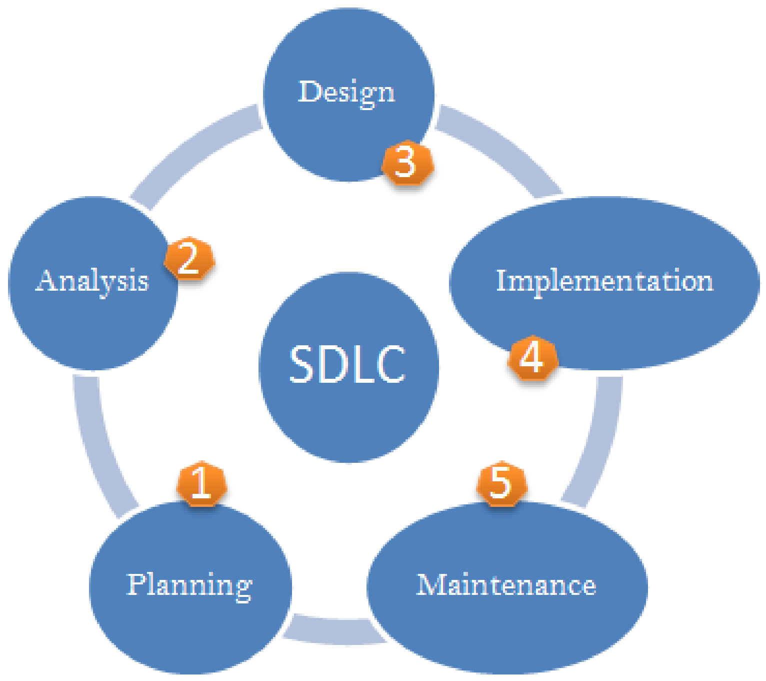
Focus on Architecture, Design and Implementation choices:

It signifies that SPE processes should be adopted at every stage in the SDLC. This entails the ‘ART’ piece of SPE and answers the question of “How & When” to implement it.

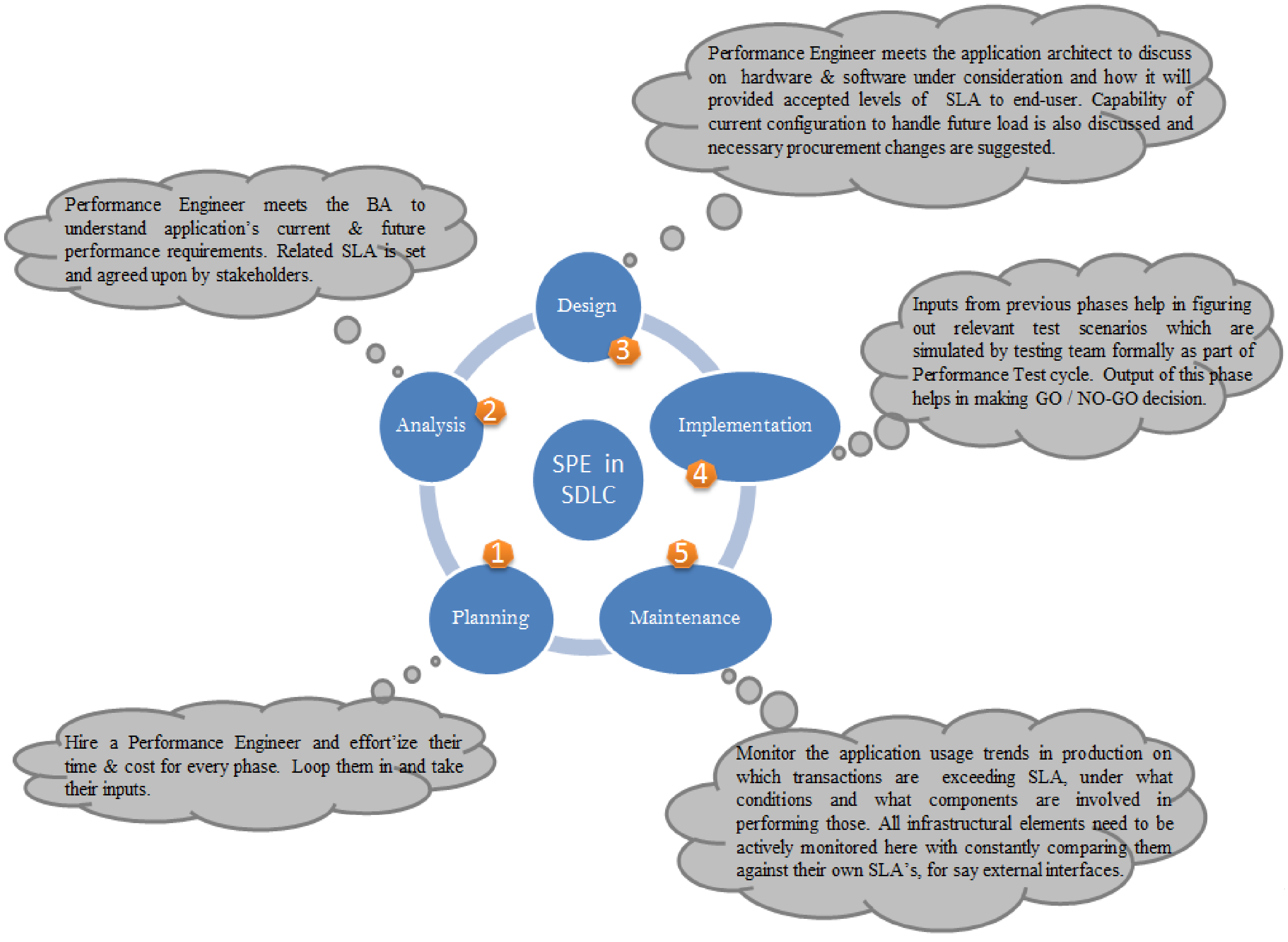
Not following this would only spell doom for the end product in no small measure.

The obvious next question is **“How do I squeeze SPE in SDLC ?”**

Answer: Depicted below is what typically followed across most of the organizations:



When SPE processes are adopted in parallel to these, what we get is a thorough *Performance Management Solution across all tiers.* What we need is *a fusion of PERFORMANCE factor into the respective Product/Service/IT Solution development life cycle, let it be, Waterfall, Rapid or even Agile and so on.*


Performance Engineer meets the application architect to discuss on hardware & software under consideration and how it will provided accepted levels of SLA to end-user. Capability of current configuration to handle future load is also discussed and necessary procurement changes are suggested.
Performance Engineer meets the BA to understand application's current & future performance requirements. Related SLA is set and agreed upon by stakeholders.
Design
3
Inputs from previous phases help in figuring out relevant test scenarios which are simulated by testing team formally as part of Performance Test cycle. Output of this phase helps in making GO / NO-GO decision.
Analysis
2
Implementation
SPE in SDLC
4
1
5
Planning
Maintenance
Hire a Performance Engineer and effort'ize their time & cost for every phase. Loop them in and take their inputs.
Monitor the application usage trends in production on which transactions are exceeding SLA, under what conditions and what components are involved in performing those. All infrastructural elements need to be actively monitored here with constantly comparing them against their own SLA's, for say external interfaces.

It should be noted that above highlighted SPE steps are iterative, meaning performance requirements & SLA's might and in most cases will need a revisit after looking at the available infrastructure which in-turn will be re-considered after post-production monitoring in Maintenance phase. Only after adopting above methodology as a continuous Product / **Application Performance Management** (APM) cycle in SDLC, any end product can sustain the customer's stringent performance expectations in the long run.

Now let's say being a project manager, I have invested (monetarily and manpower) in Performance Engineering activity for my project.

So, what next? - Stay with me and I'll give you few pointers.

Analysis Phase:

The Performance Consultant needs to work hand-in-hand with Technical Architect, Business Analyst and relevant business team members to collect detailed data about application performance requirements. Some of the questions that need to be answered in this phase are:

1. Which transactions are business-critical from functional and non-functional view-point?
2. What and When is the application peak workload condition?
    - **Workload Analysis**
3. How many types of transactions will the application/system under test (SUT) perform?

- Online Transactions (**OLTP**)
- Batch Jobs
- Background Processes
- Any other miscellaneous transactions that are expected to be processed in addition to above
- Service Level Agreements and their boundary conditions for all transactions

This leads to **Transaction Processing Analysis**.

Now that we know the performance requirements, next step is to understand application architecture, because it's not wise to fight a beast without knowing who the beast is and what its weapons are (metaphorically).

Design Phase:

Sit with application architect and discuss the following:

- Input systems to AUT
- Output systems to AUT
- Various infrastructural components involved in application design
- Technology used for core business logic
- Protocols used for Intra and Inter application communication

These queries, among others, help Performance Consultant to better understand the End-2-End architecture of application under test (AUT) with all external interface dependencies. The architecture on hand needs to be able to handle the non-functional requirements gathered in analysis phase like peak workload condition to ensure near-zero downtime operation. By blending performance factor into architecture/design review, it makes it much easier and cost-effective to plan for Hardware Sizing too.

Testing/Implementation Phase:

Now that we understand the architecture and have designed it in a way to complement performance requirements/SLA, it's time to put it all together and perform End-2-End Performance Test. It is a type of non-functional test which aims towards validating if current AUT is capable of handling expected transaction processing workload under varied conditions. Based on the requirements and constraints on-hand, following are few categories of performance tests that might be conducted:

- **Load Test**
- **Endurance Test**
- **Stress/Scalability Test**
- **Volume Test**

Prior to test initiation, it must be ensured that –

- Test Environment is an exact replica of To-Be Production Environment (Ideal Condition).
- Monitoring checkpoints are installed at every infrastructural component of transactional data flow.
- Alerts and Logs are properly setup for every component involved in testing
- Sign-off has been received from relevant stakeholders on all performance test artefacts like Test Plan, Test Strategy, Test Scenarios and proposed Monitoring checkpoints/measurements.

Please note that above list is only inclusive and not exhaustive.

Performance Testing is often followed by Tuning or Optimization activities which are in-turn followed by regression performance test cycles until the desired SLA is either achieved or changed (after in agreement with stakeholders) considering time and budget constraints.

During final Test Report sign-off, the Performance Consultant must ensure that customer understands the risk of going LIVE with current performance issues, if any, at hand and is ready to face any consequences as a result of them.

Maintenance Phase:

The application is now LIVE. This doesn't mean that Performance Consultant can shake his hands off it. The Game has only begun now, for real.

S/He must proactively ensure that **Post-Production Monitoring** techniques are in place. This enables the stakeholders to get real-time information on the performance of their application being used in the wild. This information, if contained, can be immensely helpful for the Technical Architect and Performance Consultant in future for **Capacity Planning** exercises through techniques like **Data**

Mining. Also, it helps in re-designing the Performance Test Scenarios to model near-to exact user behaviour. This enhances the efficiency of Performance Test Suite. Regardless of whether SPE processes were followed during SDLC or not, appropriate post-production monitoring often unearths the issues (and their trigger conditions) that were possibly missed either in design, coding or testing phase.

If you are thinking, "Wow…Lots of new words" – Then, Don't worry. In my upcoming articles, I'll give more details and specificity to these terms along with the tools and techniques that have proven to be handy in Software Performance Engineering arena.

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

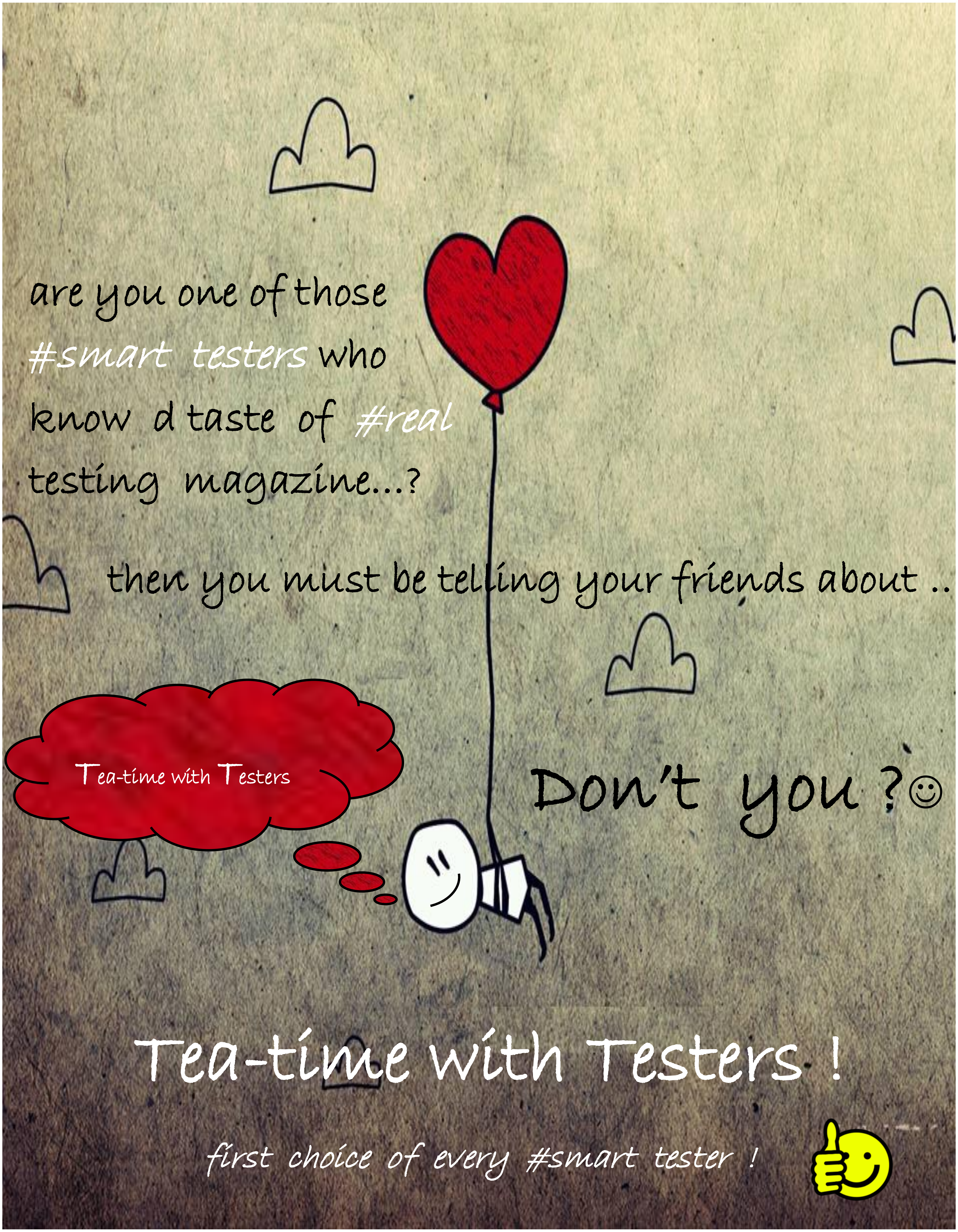
Do YOU have IT in you what it takes to be GOOD Testing Coach?

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 .

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 1

Are you a **Professional Tester?**

Chances are that if you are reading this article then you are…

And I don't mean this because I wrote this article – there are countless other testers who have better stuff to share than I do!

I mean in general, if you are reading a QA-related article in your free time in order to improve your testing skills, you fall into the small (& hopefully growing) number of engineers determined to be *Professional Testers*.

In search of the *Perfect Excuse*

Last week I saw another discussion in LinkedIn asking *"Why is testing not considered a profession?"* by many people in the Industry.

There were answers ranging from "*because testing is not formally taught in Universities*" and all the way to "*because testing is new and people are still learning how to do it professionally*".

I searched in vane for someone to come and throw the blame back at us, the testers, saying the reason we are not considered a profession is because many of us are not professionals in the way we do our work.

But I guess people were too busy been self-pitying and unjustly-victimized in order to notice that most of the blame resided on us.

Looking for the answers in the mirror

Let's be honest, wherever we are not treated as (testing) professionals it is because we have not made it a priority to behave like professional testers.

Based on my limited experience, everywhere I've seen testers taking their work seriously and striving to improve intelligently, I've also seen how they were treated with respect and how their work was appreciated thanks to the value it provided to the Organization.

So to the point:

What are the 10 main reasons? You are not a Professional Tester?

1. You think testing is *not a technical profession*, and so you don't even try to understand the code behind your product!

If you work on Software Development you should understand at least a little about software engineering.

As a tester, you need to be able to read code in order to analyze your product and understand how changes and fixes can affect it and cause additional bugs. The days of hiding behind "black box" vs. "white box" testing are over.

You can still get away without writing any code if you don't want to, but as long as you refrain from reading the code you will be missing a very important input to your overall testing process.

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"

Answer yourself truthfully: When do you start getting involved in the development process?

In theory we'd like to start during the requirements gathering and analysis phase, together with the rest of the team. But in practice we hardly provide any inputs before we are "hit in the head" by the first build from our developers looking for feedback on their features.

Why does this keep happening? Most testers will say that it is because of the "viscous-circle" of been the last link in the development chain; we are always extremely busy testing when "the others" start planning.

But in truth, if you cannot spare 2 hours a day to take part of a feature design meeting it means you are a lousy time manager. It also means that the only reason you are not part of the development process earlier is because you don't make it a priority; or in other words because you don't want to!

3. Your only interaction with a Customer is when your Support Team asks you to reproduce a bug from the field.

Part of your job description is to test your product based on the way it will be used on the field and to catch the bugs that will be important to your users once the product is released.

In fact, your job is to be your customer's advocate within the development team. To plan your tests and set up your environments based on their working behavior. You are also expected to provide functional feedback based on their needs and constraints.

If this is the case, then how can you simulate field work and represent your users if you don't know them? When was the last time you visited a user to understand how he or she uses your product? Can you really relate to the work they do with your system and with the constraints of their working environment?
I guess the answer is NO.

Go and visit some of your customers! Until you know and understand your users, you will keep doing a lousy job as a tester.

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

There are a small number of simple truths in testing; maybe the most trivial of them is that *"no tester will ever have enough time to test everything"*. This is where *Basic Risk Management* comes into play, helping us prioritize our work in order to know what needs to be tested (and tested first) and what can be assumed to work based on the results of other tests.

But as I said, this is only the basic side of Risk Management… The more advanced side of it, and one that

provides no less value to your team is the one that is not directly related to testing at all!

Every tester knows there are areas of his product that are more risky; areas where there are always more bugs and where the work of the team is always delayed due to unscheduled and unplanned circumstances.

It is part of our job as testers to be aware of these areas and remind the team about them during all stages of our projects. This way we can choose whether to develop the features using different areas of the product, or if necessary schedule more time to stabilize the system, accounting for these "unplanned issues" that will always present themselves.

You should strive to shed light on the issues, whether existing or potential, affecting your product. Helping the team to set realistic objectives and reach your goals on time and on budget.

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

The Testing Profession is in many ways uncharted territory. There are many paths that will take you into testing, and as many ways to improve professionally once you are part of the Testing World.

Most of these "testing improvements paths" are individual, and will be a mix of the individual capacities of the tester, together with the needs and constraints of his current workplace, and the information sources available to him at the present time.

In short, there is no ONE WAY to develop yourself professionally as a tester, and these improvements will not be easy or come quickly. So, unless you decide you want to seriously invest in your development process, and only after you understand how to achieve this goal, will you be able to really improve your testing skills and the value you provide to your organization.

How do you achieve this?

Start by mapping your strengths and weaknesses as a tester, then decide what areas do you want to develop (that will also be valuable to your Organization), and finally look for the means available to you to develop these skills.

One thing is certain, it will be completely impossible to improve if you leave it to chance, or to another tester to tow you along during his personal development process.

To be continued in the next issue…

I made myself a promise not to write article that are too long. So I will cut this one here and write a follow-up article later in next issue) with the additional 5 reasons why you are not a Testing Professional.

Since I don't want to leave you hanging, the next 5 reasons are around your career path as a tester, working mainly with scripted testing, not using automation as a tool, and general skill-set management.

Feel free to provide your feedback on these reasons, or even provide additional reasons why you think we are not treated as Professional Testers by our peers. I am sure each of us has something good and insightful to contribute on this subject!

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

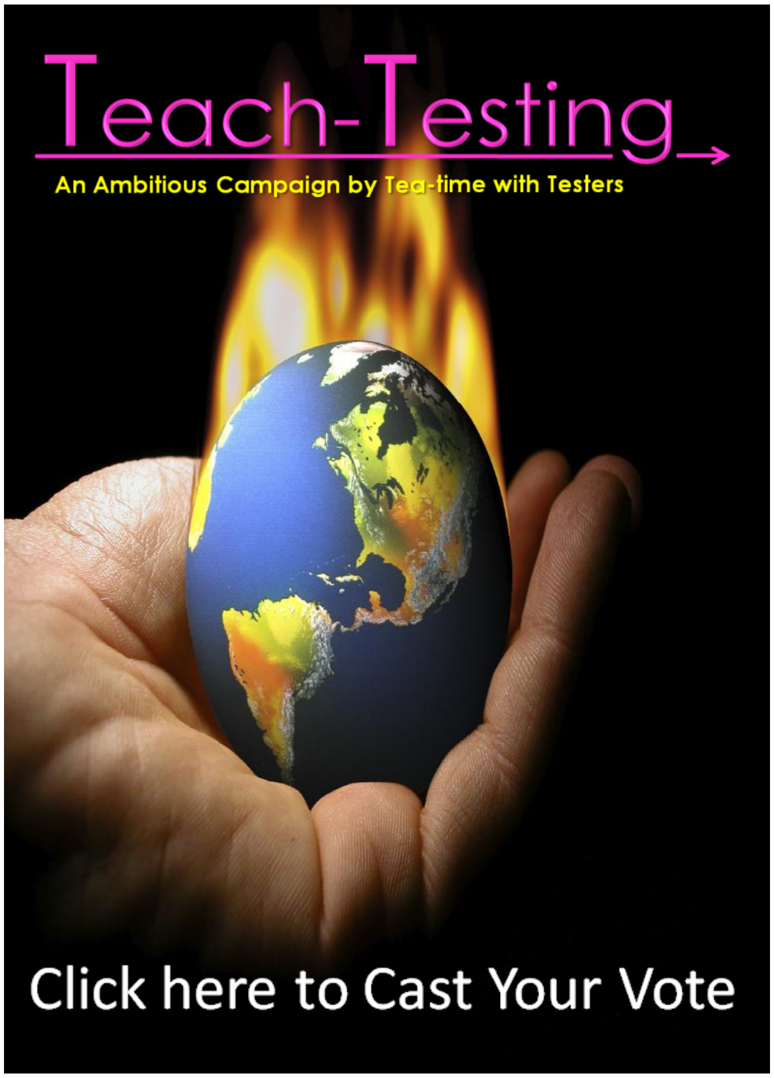
He's worked in companies ranging from small Internet Start-Ups and all the way to large multinational corporations, including Mercury Interactive (currently HP Software) where he managed the QA for TestDirector/Quality Center, QTP, WinRunner, and additional products in the Testing Area.

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

Call for Articles !

Have you got something to say?

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

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

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

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

Click HERE to read our Article Submission FAQs !

sciencephotogallery

The Promising Athlete

"What is the objective of load test?" I asked the participants attending my workshop.

"It is to find out if the system can handle the load" replied one of them. A circular answer, this does not help!

Another person chimed in "Having multiple users use the system at the same time and check if the system functions properly".

"Does that mean that load test is only applicable for multi-user systems" I asked. Silence followed.

I have observed that the clarity of understanding of "what some of the common non-functional tests like load, stress, performance, scalability, and volume" is typically lacking. Possibly, because the tests are inter-related. Possibly because the participants did not go beyond of these jargons!

Using "Joe the promising athlete" as an example I have found it is very easy to explain the objectives of these tests and then delve into the test design techniques for these.

Let me share this with you now.

Joe is a young athlete, his coach seeing in him a future world champion. Every day Joe spends significant time at the gym building muscles/strength. He lifts weights and is comfortable lifting up to 40kg snatch a few times.

The coach observing that Joe is comfortable with 40 kgs hands him 50 kg weights. Joe is successful, but finds it progressively difficult. After a few minutes he pauses to catch his breath. The coach hands him 60 kg!

"Coach" Joe whines. "Go on Son" says the coach in a baritone voice.
Joe does succeed, his body glistening with beads of perspiration.

After stating this short snippet, I asked the participants in the workshop "So why is Joe whining?"

"Well he is tired" says a participant.

"Why is he tired? I ask.

"It is pretty obvious, he is run out of energy", says another.

I continue with my next question "So why has he run out energy?" knowing fully well that my audience will think that I am pretty dumb.

"Well he has done a lot of work and that is the reason Joe has run out of energy".

"Great. So Joe has expended energy doing work. When his energy has run, he is unable to continue further" I summarized.

"Now, can you tell me how his story relates to load/stress test?" I ask.

I see the participants eyes brighten and one of them chips in "Oh Yes, Load is about subjecting the system to do "work" using the energy i.e. system resources. 60 kg is the maximum Joe could lift as he was stressed out, and therefore stress test is finding out the maximum load that the system can "lift" before the energy i.e. resources give out".

"Wonderful guys. Let us summarize. Load test is about understanding if the system can do the typical real life work with the given resources while Stress test is about understanding the maximum amount of work that can be done by consuming 'all' the resources".

Now the coach seeing Joe is tired gives him an energy drink. After a few minutes Joe is ready to go and the coach hands him 75 kg. Joe now energized, lifts it with aplomb!

"Now what can we learn from this?" I ask.

The participants think for a while and then "Well I think Joe got an additional shot of energy from the drink and he was able to lift more" said the guy in the last row.

"Wonderful correlation. Yes, he could do more work with the additional energy. Now can you connect this to scalability test?"

After a momentary pause the person on the far left says "Yes, once stressed, more energy allows us to do continue to do more work and hence scalability test is about checking if the system when stressed out can indeed do more work if given more resources."

"Brilliant. Now would one of you summarize this please" I ask.

The short guy with glasses in the first row says "Resources (i.e. energy) enables us to do the various System Operations (i.e. work). The three tests Load, Stress & Scalability" are about connecting these in different ways. I will illustrate this on the board".

He walks to the board and writes:

Energy (Resources) ---enables--> Work to be done (#System Operations)
Given a set of resource, Can I do the typical #Operations? ==> <u>Load test</u>
If I use all given resources, What is max #operations that I can do? ==> <u>Stress test</u>
If given more resources, Can I do more #operations? ==> <u>Scalability test</u>

Operations = Set of features used to accomplish a job & A typical day requires different jobs to be done by different end users of a system.

"Great summary. Indeed simple. Now how does this relate to performance test?"

"That is easy. Performance is do with time, the time taken to the work (operation). Performance test is about measuring if the time taken to perform the operations is indeed acceptable (with the given resources)"

"Does this mean that the load and performance test can be done at the same time? The objective however of these two tests are indeed different, is it not?" asks the back bencher.
"Yes, you are right, they may be done together, however their objectives are different" I say.

"Before we conclude, one last question - What is volume test?" I ask, walking to the end of the room.

After a few seconds, a quiet but firm voice behind me says "System Operations is really about processing data and volume test is about checking if the system can indeed process large amounts of data. By the way in some systems large volume can choke and stress the system".

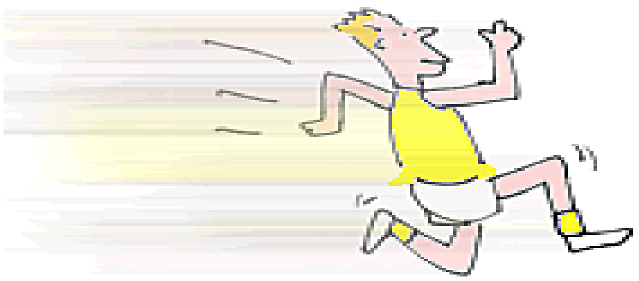
"Guys, guess Joe and his coach helped us understand Load, Stress, Scalability, Performance and Volume test. I hope you understand that any kind of system (single/multi user) can be subjected to these tests. Real life load is about subjecting the system to various types of operations (concurrently if multi-user) and not just subjecting the system to a blast of a specific operation" concluding the class.

"Yes" chimed in the participants with a big smile.

Thank you guys. I am done with the class.

It is Tea Time now!

Enjoy.



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** .

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.

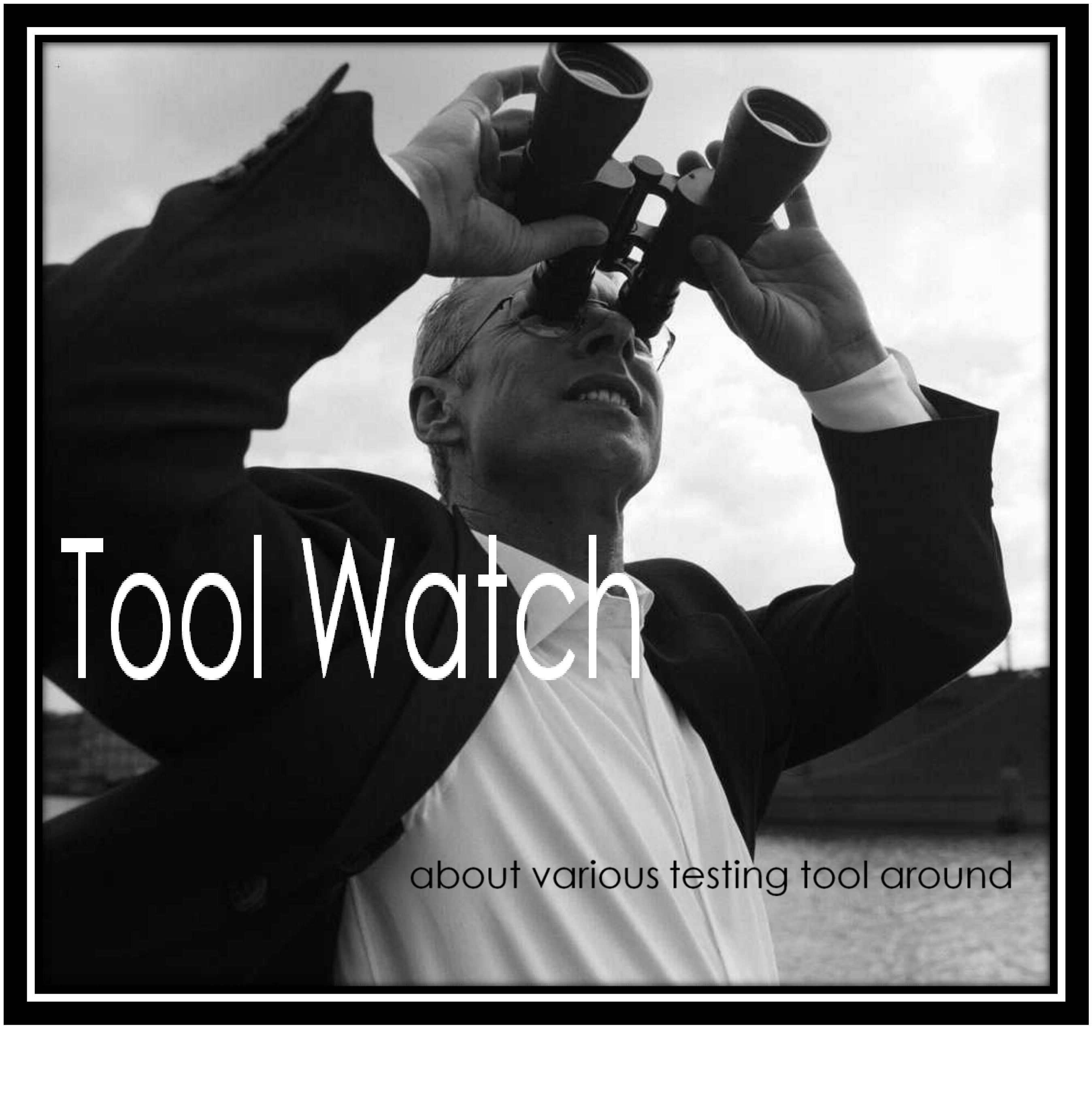
Quality Testing also provides daily Polls and sample tests for certification exams, to make tester to think, practice and get appropriate aid.

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.


Tool Watch
about various testing tool around

Rapid Reporter

PART 4

By Chris Philip

❖ Aggregate many sessions into one single file

After you've done a few sessions, you may want to consolidate all the *.CSV files into a single one, to allow analysis of the whole project to date together.

To do that, run the following command in the command line, which will concatenate *.CSV files into a single one. The command will take any file with a file name ending in the ########_######.csv format (*where '#' represents a digit*), without validating the content. Consolidated reports files (*which start with "report_"*) or files with a different name format will be ignored.

```
C:\> RapidReporter.exe –report
```

❖ Arrange persistent data in order to track test results

There are three ways to use persistent notes. In order of preference:

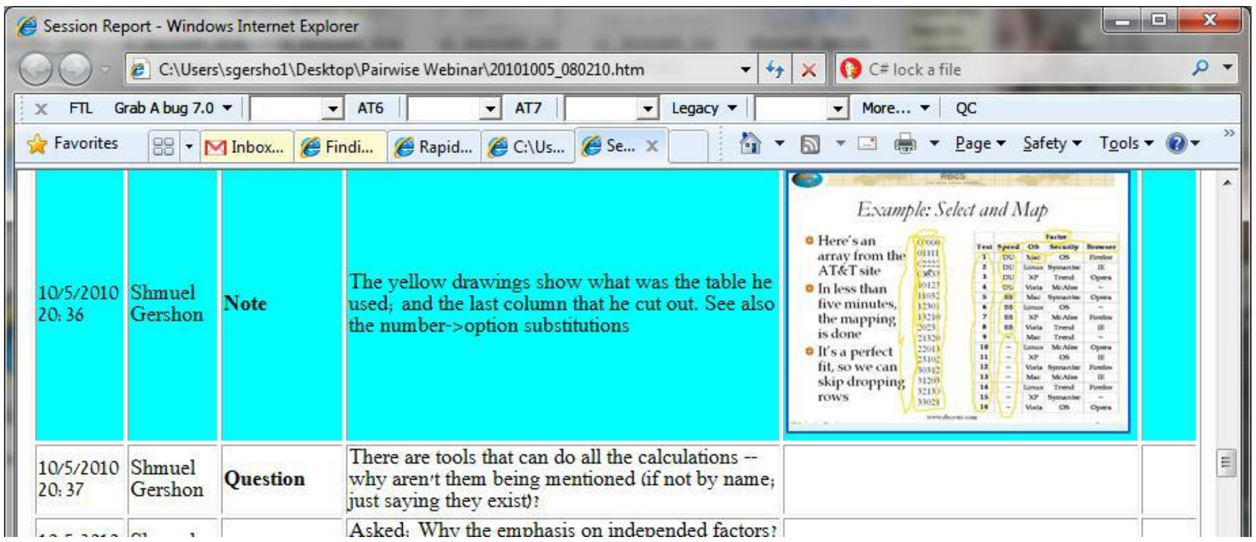
1. Use the extended notes screen. These notes will stay on until you delete them or press the clear button. This area is good for keeping track of information over many notes (*even creating incremental attachments*).
2. Use the right-click context-menu in the notes area. This will show you all your past notes.
3. "Open working folder" from context menu and look at the notes directly from the *.CSV file. This is not recommended because some apps will lock the file access, which would cause an error message to appear.

Different Usages

You can use Rapid Reporter for other activities.

1. One good usage of Rapid Reporter is to listen to webinars.

a. Start Rapid Reporter with a smaller set of note types, like "RapidReporter.exe Information Question Lookup". Then, jot your notes of what the speaker is saying on the application, and your questions. Whenever there's an interesting thing on the slides on screen, take an automatic screenshot to accompany your notes. You can later print all the notes and screens in HTML.



10/5/2010 20: 36	Shmuel Gershon	**Note**	The yellow drawings show what was the table he used; and the last column that he cut out. See also the number->option substitutions	Example: Select and Map Here's an array from the AT&T site In less than five minutes, the mapping is done It's a perfect fit, so we can skip dropping rows	
10/5/2010 20: 37	Shmuel Gershon	**Question**	There are tools that can do all the calculations -- why aren't them being mentioned (if not by name; just saying they exist)?		
			Asked: Why the emphasis on independed factors?		

2. A similarly useful way to use Rapid Reporter is to take minutes, or meeting notes.

a. Start Rapid Reporter with a set of note types that represent participants, as in "RapidReporter.exe Moe Larry Curly". Then you can write each person's contribution, which will be time stamped and organized for you. If the meeting has online slides, you can add the slides to each contribution by taking a screenshot.

Back To Index

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

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

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

* Conditions Apply

Chris Philip has been working with his first employer TCS since 2 years, passionately in area of Automation Testing. The passion and interest in automation was revealed during his college projects in robotics and successful completion of project work from India Space Research Organization, automating the microscopic and sensitive calculations regarding the minute changes in accelerometer data in launch vehicles, rockets and spacecrafts. The project was done in microprocessor programming language.

Chris is active member of Linux club, IEEE and Computer Society of India (CSI).

His special interests are software automation, having extensive hands-on experience in QTP, Sahi, Selenium and RFT.

Actively participates in blogs and discussions related to automation practices. He has presented 3 white papers till date about the automation practices, short cuts and interesting logics applicable in Quick Test Professional.

Chris is reachable at **christhomsonphilip@gmail.com** & on twitter **@chris_cruizer**

by

by Sebi

Claim your Smart Tester of The Month Award. Send us an answer for the Puzzle and Crossword bellow b4 15th April 2012 & grab your Title.

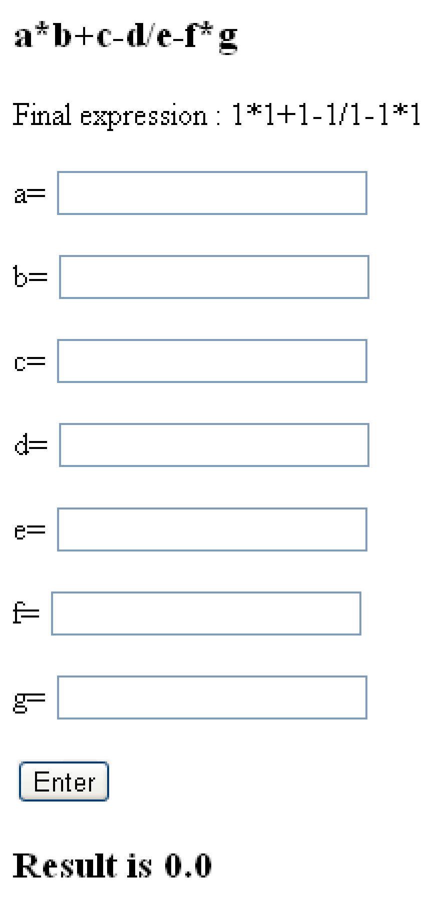
Send -> teatimewithtesters@gmail.com with Subject: Testing Puzzle

Gear up guys.......

It's Time To Tease your Testing Bone

Puzzle "Play with the Patterns"

What is the biggest bug that you can find for this application http://testalways.com/eval/ ?"



a*b+c-d/e-f*g

Final expression : 1*1+1-1/1-1*1

a=

b=

c=

d=

e=

f=

g=

Enter

Result is 0.0

Enter the input values for the expression and check if the result shown is correct

Biography

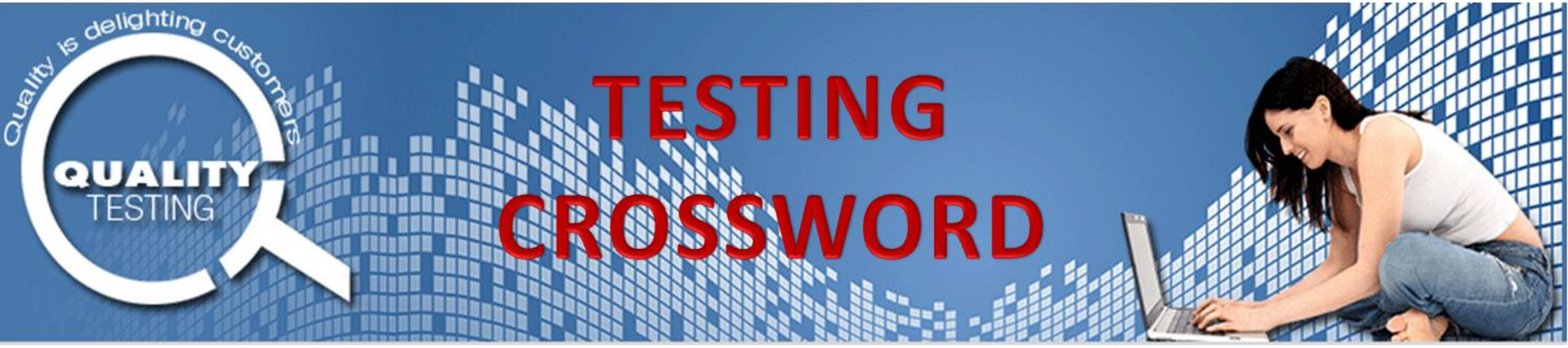
Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years. He is currently hosting European Weekend Testing.

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

He tweets as @testalways.

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

Back To Index



TESTING CROSSWORD

1		2		3		4			5
	■		■		■		■	■	
6					7				
	■		■	■	■		■	■	
8		9			10		11		
	■		■	12	■		■	■	
13				14			15		
	■		■		■	■		■	
16				17				■	

Horizontal:

1. _____________Testsuite, has been named "Best in Test 2012" for Functional Testing (5)

6. It is a GUI test automation tool for Adobe Flex applications (7)

7. A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it (4)

8. Performace Unite Testing Tool (5)

10. A formal product evaluation performed by a customer as a condition of purchase, in short form (3)

11. A program or test tool used to execute a tests, in short form (2)

13. ________ offers free crowdsourcing services for uberSVN Users (5)

14. It is a tool to enhanced wiki and issue tracking system for software development projects (4)

15. Code Coverage, in short form (2)

16. A testing phase where the tester tries to 'break' the system by randomly trying the system's functionality (5)

17. It is a common, relatively simple file format that is widely supported by consumer, business, and scientific applications (3)

Vertical:

1. It is the test management tool for agile projects (9)

2. Running a system at high load for a prolonged period of time is called ______ testing (4)

3. It is a test automation platform that greatly increases productivity and drastically reduces maintenance overheads for test automation (3)

4. The behavior produced/observed when a component or system is tested, first word (6)

5. Checks for memory leaks or other problems that may occur with prolonged execution is called_________testing (9)

9. It is the world's largest professional association dedicated to advancing technological innovation and excellence for the benefit of humanity (4)

12. A group of people whose primary responsibility is software testing, in short form (3)

15. Capability Maturity Model, in short form (3)

Answers for Last Month’s Crossword:

T	A	R	A	N	T	U	L	A	M
E	■	A	■	T	■	N	■	■	O
S	Y	N	T	A	X	I	M	T	N
T	■	D	■	■	■	T	■	■	K
I	S	O	L	A	T	I	O	N	E
N	■	M	■	■	M	■	R	■	Y
G	A	L	L	I	O	P	A	T	H
B	■	■	O	■	D	■	C	■	Q
O	D	T	O	M	E	T	L	U	A
T	■	■	P	■	L	■	E	■	T

Answer for the last puzzle : **1**

We appreciate that you

“LIKE” US !

Join us on Facebook.

You are just a CLICK AWAY

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

Our Testimonials

Superb magazine and smashing articles !

I would like to thank each and every person who is involved in this.

- Prashant Arakeri.

My Voice on "Teach-testing"

Looking at present state of fake and money minded trainers, training institutes, I strongly recommend teaching software testing in colleges.

Since naïve students and young professionals don't have enough knowledge about software testing, certification factories are just milking them for money. This should stop somewhere.

- Ashwini Chougule

I see other testing magazines (even other Indian magazines) which are filled more with useless ads and promotions than useful testing related content.

I admire Tea-time with Testers for value it is adding to the community and their endless efforts. You are the example of ideal magazine..no I guess ideal publication.

Keep Rocking!

- Sujit Verma

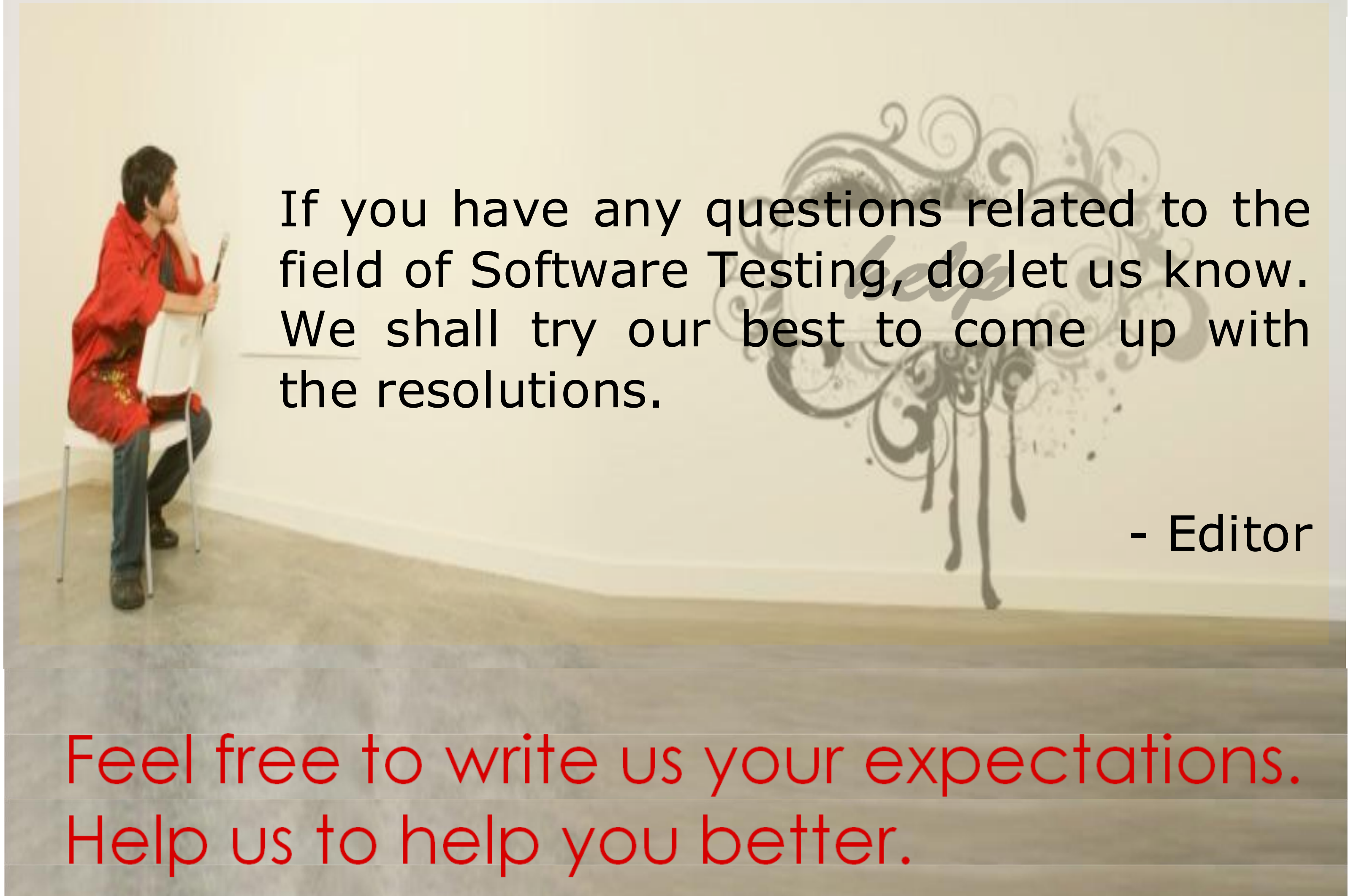
Highly impressed by the line up of articles and your writing crew. I'm amazed to see the way you people actually manage everything , every month.

Hatts off ! Indian testing community should really feel proud of you guys.

- Neha Mujumdar

You ask...

We'll help...!



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

- Editor

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

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

in next issue

articles by -

Jerry Weinberg

T Ashok

Joel Montvelisky

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

Google

Join our community on

facebook

Follow us on

www.teatimewithtesters.com

Give Feedback