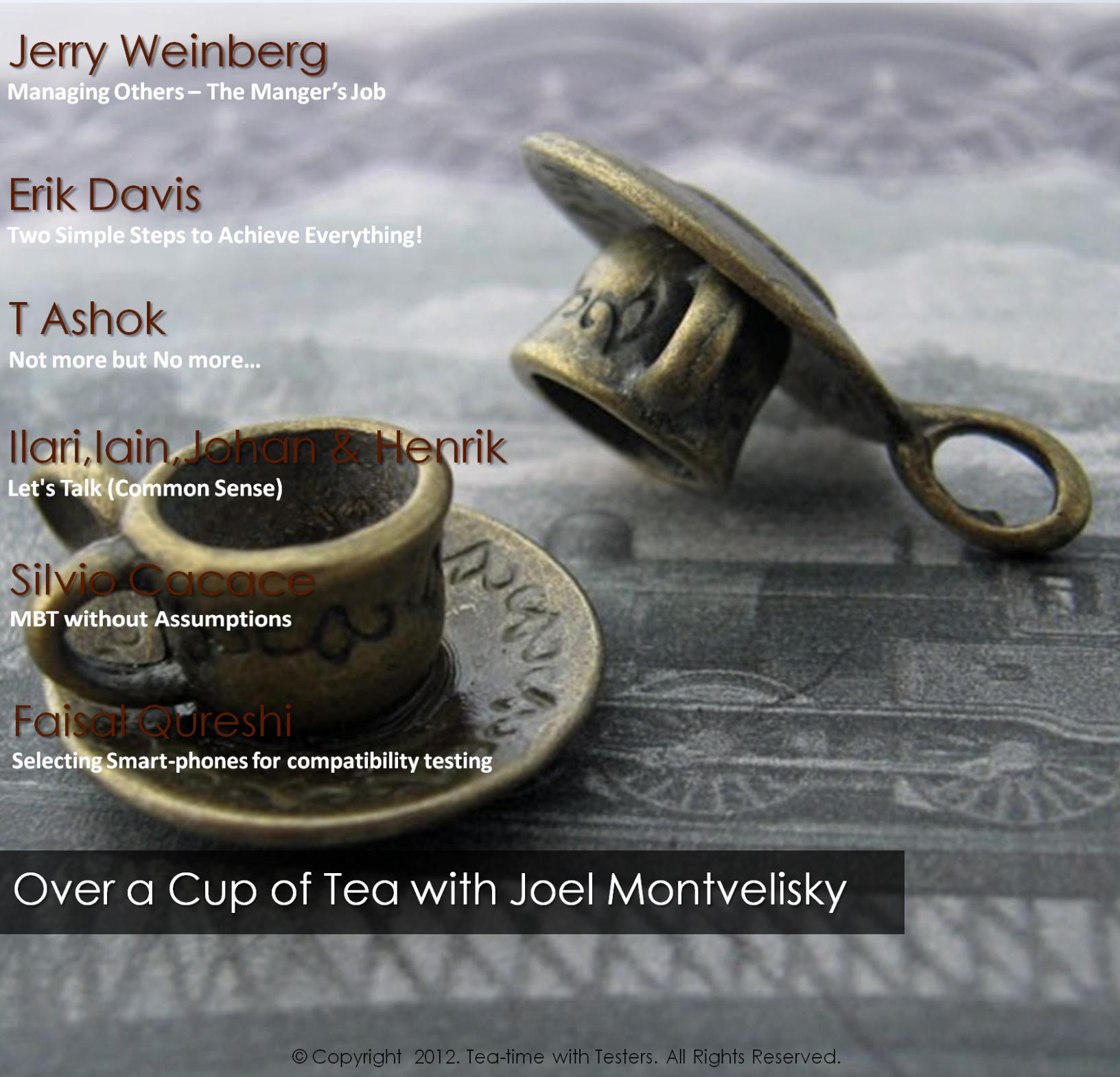
THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

AUGUST 2013 | YEAR 3 ISSUE VII



© Copyright 2012. Tea-time with Testers. All Rights Reserved.

QAzone presents

Rapid Software Testing by James Bach

December 10-12, 2013

New Delhi, India

~ Lucky Student Contest is now OPEN ~

Participate to grab your **FREE** seat

Click HERE for more details

TEA-TIME WITH TESTERS

First Indian Testing Magazine to reach 101 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) 8275562299

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 golden circle of Tea-time with Testers

I must thank Leah Stockley for suggesting this awesome TED Talk by Simon Sinek, the other day.

In this presentation, Simon talks about his golden circle theory which explains how great leaders and organisations inspire action.

We (team TTwT) have always made efforts to make this magazine a medium to inspire and promote great actions in testing community.

Simon's talk made me wonder if Tea-time with Testers has ever succeeded in inspiring testers to do great job, in inspiring them to get serious about testing skills, in inspiring them to contribute to this changing face of software testing. Though my heart was saying 'yes', there was no concrete evidence to prove it to myself.

I reached out to some leaders and experts in community to find out what they feel about TTwT, what makes them read this magazine, why they advise others to read it and I got these answers.

That made me happy but….not completely. Why so? My mind made me ponder over it again. Okay, some great leaders and testing experts (celebrity testers in common man's language) find this magazine good but what about those thousands of readers who are not celebrities? And to find that out, we launched this Ideal Reader Contest. There are still odd 10 days for this contest to be over but we are happy with whatever responses we have received so far. I felt overwhelmed after knowing why people read Tea-time with Testers…and somewhere it assured me that, TTwT ***has*** some contribution towards inspiring better things in community.

We will be declaring the result of 'Ideal Reader Contest' in next issue but before that I want to thank all who participated and gave us their valuable feedback.

With everything we do, we publish and we promote, we believe in challenging the status quo, we believe in challenging (so called) best practices. The way we do it is by publishing real life stories, insights and experience of testing improvements, by our visually appealing design, unique ideas and initiatives. And we just happen to create magazine that satisfies its every reader!!!

And that explains the golden circle of Tea-time with Testers.

Enjoy Reading!

Lalitkumar Bhamare
editor@teatimewithtesters.com

WOW!!!!!

Yes, 'Wow' was the very first word that came to my mind when I first read your magazine. It has been just a random google search on 'good testing articles' and when I came across 'Tea time with testers' , I could not resist myself from reading it and no surprise in saying that I got totally addicted to this magazine within just 2-3 days of time. Whenever I have some free time in my work, no matter even if I get only 3 min of time to be free from my work, I read the articles written in your magazine.

I felt really happy and satisfied after reading your magazine and feeling proud to say that I am really trying to follow the instructions/guidelines written by many senior testers. As being a tester, I need to be very creative, active and smart - this is what I had learned and many more things from 'Tea Time with Testers'! I would surely recommend TTWT to all my friends and colleagues.

I wish Lalit and his entire team all the very best and looking forward to more and more interesting articles :-)

Last but not the least, I truly love the articles/facts written by Lalit in Editorial column.

Thanks,

Sri Bhargavi

Software Engineer

And a lot more is posted on our Facebook wall.....

I read 'Tea-time with Testers' eagerly, becoz ...Cafe Coffee Day says : "A lot can happen over coffee". I will say :"A lot (Loading Outstanding Test ideas) can happen over TTwT" **- Christi Henitha**

I love to read Tea-time with Testers. I get good testing knowledge @ one place i.e. @ TTwT. And last but not least, I am a software tester and this makes me smarter!! **- Kapil Saxena**

TTwT is the bet "Guru" for providing the actual knowledge with the valuable experience of other tester. Many times I have got the solutions from my bottle neck position while testing. **- Zankesh Jain**

TTwT magazine plays a vital role in enlightening my approach towards software testing. It provides excellent articles and latest trends in testing. **- Maheshkumar Jaghamani**

I read 'Tea-time' to learn and to enjoy, read to enhance my software testing knowledge, to affirm what I know and to experience the joy of exploration. **- Kay Mak**

I read Tea time with Testers because it is full of fun and gives brilliant ideas for gaining knowledge and provides best guidance about how to do testing. **– Mahesh Kumar**

Tea-time with Testers is the most popular Software Testing Magazine that I have ever known! **– Varun Sundar**

I read Tea time with Testers, because it gives immense knowledge about different aspects of Testing....Mainly to gain more knowledge and improve testing ability....

- April - May month issue of TTwT which published the article series about 'Addressing the Risk of Exploratory Testing' - was superb, and it helps us lot to improve our testing ability in exploring..... **– Wasif Ahmed**

QuickLook

Crossword by

Testing Puzzles

by Sebi

Singapore Testing Forum

Leah Stockley —August 25, 2013

I'm always reading about the numerous Tester Meet-ups and testing forums/ conferences that are happening around the world, and was disappointed there was nothing available for us testers in Singapore. So I decided to stop complaining and set one up myself.

My first foray was kindly sponsored by the AST (http://www.associationforsoftwaretesting.org/) in December 2012. You can read the write up here. It was a great event and the turn out confirmed my view that the Singapore Testing Community wanted a regular opportunity to meet with others in the industry on a regular basis. So I set about seeking some sponsors to help me set up a regular meet up.

Step up The Testing Consultancy and Tricentis, people who share my passion that Testing is an intelligent activity to be done by skilled individuals. Together we set up the Singapore Testing Forum.

Our first event was held on June 18th. Once again we had an encouraging turn out of more than 80 testers from various companies and industries. We started with a few words from our sponsors whilst everyone tucked into the coffee and snacks they had provided! I then presented my introduction to Context Driven Testing: The rise of the thinking tester. My thoughts on how I have seen the testing industry change for the worse in the last 5-10 years and how I'm happy to be involved in a community who believes very strongly that we need to return to the days of common sense, intelligent testing.

As you'll see if you watched the video, we had a late change to the program as our planned speaker Rajesh Mathur had last minute flight issues and was unable to get to Singapore. Thankfully, Nicky Watson, a Senior Testing Manager who also shares a passion for promoting the cause of intelligent testing, stepped in at very short notice and delivered a great presentation giving tips on how to talk to stakeholders (non-testers) about Testing.

The feedback from the night was very positive. So much so, STF 2 has been organised for Sept 9th. To be held again at NTUC building. We're honoured to be flying in a guest speaker from New Zealand. And I will be continuing my series, diving into more detail to explain how Context Driven Testing looks on real projects. This time I'll be giving a demo of 'Dynamic Test Analysis', showing how testers can use Heuristics to quickly break down a system and produce a Visual Test Model to clearly communicate the understanding, or knowledge gaps, to the project team.

I hope to see all you Singapore based testers at the next event. Please join our LinkedIn group for more details of how to register.

Dates & speakers are already being set for the final 2013 STF in November and we'll be sure to continue the series every 2 months throughout 2014.

Read more…

Back To Index

International Society for Software Testing

ISST is a testing society for testers who are serious about their craft!

We're looking to change the testing world by promoting an approach to software testing that emphasizes value and the role that skilled testers play in its delivery.

And you are invited.

Join us!

http://commonsensetesting.org/join

INTERNATIONAL SOCIETY FOR SOFTWARE TESTING

Managing Others – The Manager's Job (Part 1)

When one does not know how to convince, one oppresses; in all power relations among governors and governed, as ability declines, usurpation increases. - Madame de Stael

The relationship between the Lone Ranger and his "faithful Indian companion," Tonto, remains a mysterious one.

Nobody knows what Indian language "kimosabe" comes from, nor what it means. We don't even know what "companion" implies in English. What we do know is that the Lone Ranger and Tonto seemed to work well together.

All their projects succeeded, and no commercial revenues were ever lost because the villains weren't delivered on time.

As we've seen, quality management requires that you manage yourself well, but that's not sufficient. To build and maintain large-scale information systems, you must also succeed in managing your relationships with others.

Poor management is not confined to software engineering, nor even to our moment in history. Poor management will always exist when managers are incongruent, for, as Madame de Stael said, "as ability declines, usurpation increases."

Thus, as you climb out of the technical tar pit onto the management ladder, your own feelings of inadequacy can make you into an oppressive manager. And oppressive managers may work alone and shoot silver bullets, but they will never become Lone Rangers.

A manager is responsible for

- Deciding what is to be done and appointing someone to do it.
- Listening to why it should not be done at all, why it should be done by someone else, or why it should be done in a different way than what you've prescribed.
- Following up to see if the job has been done correctly, discovering that it hasn't, and listening to some of the world's worst excuses from the employee who should have done it.
- Following up again to see if the job has been done, only to discover that it has been done incorrectly, but deciding that you'd better leave it as it is because it's as good as you're likely to get.
- Wondering if you ought to get rid of the person who can't seem to get the job done correctly but deciding that the successor is most likely to be just as bad—maybe even worse.
- Considering how much faster and better the job would have been done if you had done it yourself; reflecting that if you had done it yourself, the job would have been complete in 30 minutes, but that instead you took three days trying to figure out why it took somebody else two weeks to do it incorrectly.

- from an anonymous photocopy

The quotation that begins this series is an anonymous photocopy that has been circulating in offices all over the United States for at least ten years. While meant in jest, it seems to subconsciously support some negative stereotypes, without being funny enough to shatter them. The writer (hopefully tongue in cheek) seems to be espousing the incongruent management style that often arises from the One-Dimensional Selection Model. This is the style most prevalent in software engineering organizations, so I decided to use it as the outline for describing the manager's job.

Earlier in this series we discussed why congruence is essential in management, which apply to all individuals, whether or not they have the job of manager. This series will address the manager's congruence in interacting with other people. It is the nature of these interactions that defines the software engineering manager's job. I only hope this humorous bad example makes a good outline.

Deciding and Appointing

A manager is responsible for deciding what is to be done and appointing someone to do it.

This view describes a hierarchic, autocratic Pattern 2 (Routine) style of management. In contrast to this model, the Pattern 3 (Steering) model says that the manager's job is getting more people involved (and getting people more involved) in decisions about what is to be done, and in doing it.

Piling on

Pattern 2 organizations can be described by the nursery rhyme about the little girl with the curl in the middle of her forehead: When she was good, she was very, very good, But when she was bad, she was horrid.

The horrid part of Pattern 2 organizations is the breakdowns in control. Broken down projects run on endlessly, full of cost and frustration, producing nothing.

One sure sign of a control breakdown is to watch the rats trying to leave the sinking project ship. Unlike rats, however, most people do not actually leave their jobs until the crisis has passed, one way or the other. Instead, they sit on the edge of the deck, covering their rears and their expenses, just in case something should change to their advantage. For crisis recovery, a manager has to mobilize these sidelined people, or the resources won't be adequate to keep the ship afloat. Of course, mobilizing them in advance may well prevent the crisis in the first place.

As an organization moves deeper into crisis, the best-informed people tend to become overloaded. This tendency to "pile on" is particularly strong in the work of fixing faults, because the Pattern 2 (Routine) managers don't want anyone but the best-informed people to modify the critical parts of the software.

Figure 1 shows how this innocent and reasonable appointment policy produces overload, and eventual burnout, of the most knowledgeable people.

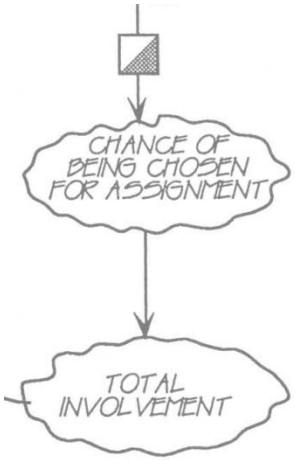


Figure 1. The tendency of managers to choose the most knowledgeable people for new assignments leads inevitably to overload and burnout. As the gray and white box indicates, managers could choose not to do this, but they often do.

Reversing the appointment policy

Figure 2 suggests that a manager can counter this piling on effect by

- choosing the least knowledgeable people for assignments
- choosing your least loaded people for assignments.

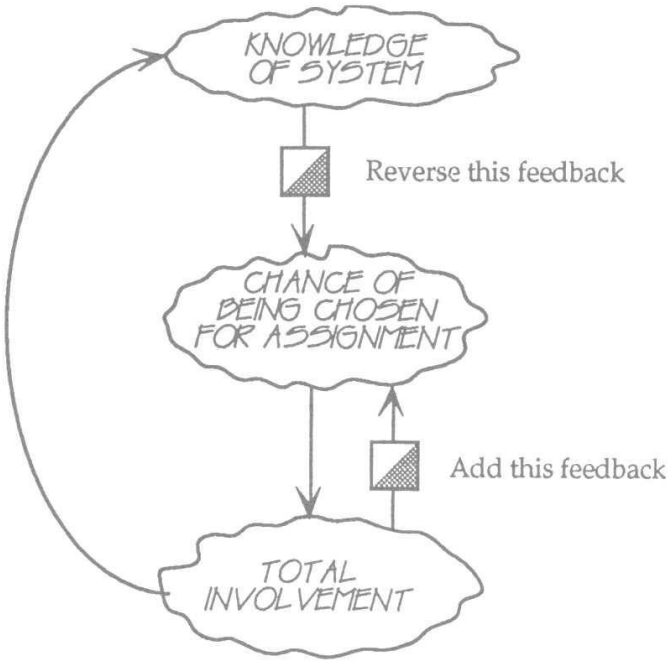


Figure 2. To prevent piling on the best workers and to get more people involved, as the manager you must reverse certain tendencies when making assignments. Assignments must be given to less knowledgeable people and to people who are not so totally involved--that is, the manager must choose the black half of both decisions in the diagram.

But can a manager really do this when in the situation's in crisis? That's how congruence comes into play.

First of all, managers must have faith that although some workers may not have experience, they do have *talent*. (If they don't have either, why haven't the managers gotten rid of them?) At the very first hint of a crisis, managers should start doing what they should have been doing all along—giving new people problems to solve as learning experiences. Managers must start this process as early as possible, and must be unflinching in their assignments—that is, do no placating. Of course, if the managers must have in place a working process for knowing what assignments are available.

Piling on the best people is not just management's tendency, but influences every person in the project. When workers have a problem, they ask themselves "Who should I ask about this?" Then they give the answer that's modeled by their managers: "The most knowledgeable person, of course." Moreover, the technical experts themselves are delighted to be acknowledged as experts, and so collude in the process. That's why managers need to establish structures to prevent the positive feedback loop of Figure 1 from occurring. In effect, they need to isolate the more experienced people from the less experienced.

To be continued in next issue…

Back To Index

Biography

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

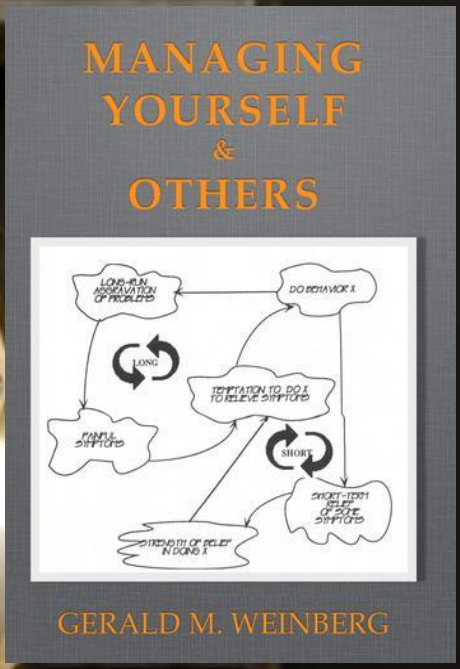
For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including The Psychology of Computer Programming. His books cover all phases of the software life-cycle. They include Exploring Requirements, Rethinking Systems Analysis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

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

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

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

MANAGING YOURSELF AND OTHERS is 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.

TTWT Rating: ★★★★★

Its sample can be read online here.

To know more about Jerry's writing on software please click here .


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

Do you enjoy whining and complaining about your life/spouse/job/boss/project/commute/etc.? If you answered yes, you and I aren't going to be very good friends for the next few paragraphs. It's OK though; I've been there. We can hug and make up at the end.

Seeing as I've been in several leadership positions at my current employer—that, and the fact that I've been here for over 10 years—I tend to have a good number of people bringing me their issues. Now, I have nothing against a good venting session; they can be cathartic if done properly. I've been told I am a good listener, partly I believe because I can let the crazy things people spew out while frustrated float right past me. The issue I have is when the venting goes beyond the point of just blowing of a little steam.

Whining/complaining/bitching about something can be very damaging. Your negative attitude will bring down those you share it with, whether they like it or not. The larger problem I have with it, though, is what are you accomplishing by complaining? As far as I can tell, usually a big fat nothing. Is that what you are looking for? Oh, it is? Well...OK, I guess you can just skip on to the next article then. Thanks for stopping by.

What's that? Your point **ISN'T** to accomplish a big fat nothing? Oh, well then, I got have a solution for you. I've developed a simple[1], two-step system to accomplish pretty much anything.

Step 1: Give a damn.

Yep, you read that right. The first step to accomplishing almost anything is to give a damn about it. Sure, you can mow your lawn if you DON'T give a damn about it. But do you think you are going to do a good job at it? Probably not. Your lines will be all wonky, you'll miss spots, and your edges will be atrocious. Start giving a damn about it, and you'll think about slowing down (or buying a better mower). "Wait a second," you're saying, "that didn't make anything better. All I'm doing is thinking about doing something different." Ah, yes, very observant of you. We are only one step in so far. We'll get there.

What about testing? Do you really think you properly test anything if you don't care about what you are doing? Sure, you may find some bugs, but if you're heart isn't in it, you'll get distracted, or just give up far sooner. Start giving a damn about your career, and you'll begin to wonder if you could be doing better. Maybe you'll think about writing up bugs better so more of them of them get fixed, or maybe you'll think about attending a conference or a local meetup.

Different people will need different amounts of time for the Giving-a-damn step to sink in and fester before you move on to step two. It could take seconds or minutes. It could take weeks, months, or even years. Pay attention to that voice in your head. If you really care about something, let that voice speak to you.

So, you've got this nagging thing in your head that you REALLY want something to be better. Maybe you've been telling people, "We really need someone to just take control of <insert meaningful item here> and get it moving." Sounds like you give a damn. Good. On to step two, shall we?

Step 2: Prove it.

Simple[1] eh? Let me explain. Giving a damn about something and telling people **WITHOUT** doing something about it is just complaining. Slightly more frustrated complaining, possibly, but it's still just complaining. So, are you going to keep whining and accomplishing nothing, or are you going to go out there and **DO** something?

You must use these steps together. As stated before, giving a damn without proving it is just French for complaining. We don't need more whiners. You also **MUST** give a damn if you are attempting to prove something. If you don't, you will likely lose interest quickly and give up.

What's that? You don't believe this can work? Well then, how about some examples?

- Fed up with the way testing certifications are pushed on testers and organizations with wild claims and no data to back it up? **Keith Klain** was, and he **did** something about it.
- Think adults should be allowed to talk like adults sometimes? **Ilari Henrik** sure did, and he **proved it** too.

- Want to help disadvantaged adults learn skills that can get them a decent job in IT with a future, oh, and maybe turn some of them into kickass testers too? There's **Keith** again, **proving** he gives a damn.
- How about this **guy**? Did he give up when the first iteration of his tester **meetup** fizzled? No. He kept at it until he got a group going that is regularly attracting a solid core of testers in his area.
- Wish there were something outside of the existing, accepted training systems to help raise the skill level of testers? **Matt Heusser** did, and he **proved it**.
- Have you ever met/heard/read about **James Bach**? Do I even need to explain how/why he gives a damn about so many things?
- How about a **tester** who, despite her nearly **overwhelming anxiety**, started a **meetup** in Michigan that drew 30+ attendees to the first event?
- What made someone as young as **Lalitkumar** start **Tea-time with Testers**?
- What if you think kids, especially those with special needs, could be represented better at the administrative level in your district? What do you do: sit in your classroom and continue to be annoyed (give a damn), or go through the required administrative training (prove it) so you can get yourself into a role where you can effect change? (Sorry no links for this one. But trust me: it happened. I married her.)

Still not convinced? I promise you, despite some of the people above being "big names," they are just people like you and me. The big difference is they gave a damn about something, and they did something to prove it.

Want me to keep going? Sure thing there, **boss**.

One last (more personal) example:

Did you just wake up as a tester (or test manager) with years of "experience" only to find you really don't know much about testing? Don't get too down on yourself; it **happens** to the best of us. The real question is what are you going to do about it? Are you going to sit there and feel sorry for yourself, or are you going to prove that you can change? Here is what I did when I found myself in this situation (all since August 2012):

- Attended CAST 2012
- Started following key people (now over 400) in the testing industry on Twitter
- Started reading numerous testing blogs
- Started running tester games at work (the dice game, Art Show, **Zendo**, Set)
- Started facilitated tester discussions at work (**K-Cards**!)
- Started a **blog**
- Got Matt Heusser and Pete Walen to come onsite to teach an Introduction to Exploratory Testing and provide a couple of days of consulting

- Gave a lightning talk[2] at the first ever **MMTMD** (Mid-Michigan Tester Meet Down)
- Started the NOTiCE tester **meetup**
- Put together two teams to compete in the NRGGlobal tester competition run by Matt Heusser
- Convinced Paul Holland to bring Rapid Software Testing to Cleveland of all places, so that I (and others in my area) could learn from his experience and the RST material
- Registered for BBST Foundations (fall 2013)
- Submitted a talk that was accepted for CAST 2013
- Registered for **CAST 2013**
- Registered for **TestRetreat**
- Will participate in the AST Leadership SIG TLC event the day after CAST 2013
- Bugged various members of the **Miagi-Do** school of testing until someone agreed to challenge me at TestRetreat
- Oh, and how could I forget: wrote this article for **Tea-time with Testers**

Please don't take this as an attempt to brag about my accomplishments. I'm not trying to sound better than anyone but myself. A year and a half ago, **none** of these things was even on my radar, let alone anything I was thinking of doing "someday." This is my attempt to show that over the past year, I started to give a damn about my career and education and about the growth and opportunities available for those around me.

"But wait," you say, "the thing for which my damn is given is far too large for me to do alone. Surely, your simple[1] two-step system won't work."

Poppycock, I say.

Let's revisit Keith Klain's participation in **Per Scholas.** If you know Keith, you know he is determined, strong-willed, and able to stand up against almost any odds. If he tried to do something along the lines of the Per Scholas **STEP** program on his own, I'm sure it would be something worthwhile in the end. Instead of trying to do it all himself though, he used his vast network of contacts to bring others on board to make the program the best possible tester training program out there, in my opinion.

Now, I will tell you that giving a damn about something and proving it to yourself and others **DOESN'T** magically give you skills in something you weren't skilled at previously. Does this mean you can only use the system in situations where you have the skill or ability to accomplish the task at hand? My short answer is no. My long answer follows:

In a previous position, I was on a committee focused on managing a specific process. Over time, we found ourselves spending an ever-increasing amount of time discussing the limitations of the current system in which the process lived. The complaining started to wear on me, month after month. Eventually I got fed up enough with the "going nowhereness" of these discussions that I announced that I was taking ownership of the project to migrate to...er…build a new system. Did I have the skills to build this system on my own? Nope. Did I dive in anyway? You betcha. So I found a team of people to work on the project under my direction.

I ran this project team for months, relatively successfully as far as I could tell. After some time—it may have been close to a year, I don't remember exactly—I began to realize that the manager of the employees doing most of the work on the project would be a better fit to run the project. He had better connections to the other teams involved in the eventual implementation. He had visibility into the non-project work assigned to the employees working on the project. He had knowledge of the infrastructure involved in running the soon-to-be system.

So I handed it off. And now the new system is in place and better than I could have imagined.

Did I fail? I don't think so. Did I do something wrong? Possibly. Should I have just not touched the project if I wasn't the right person to see it to the end? I say no. Sure, I felt a bit defeated at the point when I realized I wasn't the best person to see the project to completion. But with hindsight, I can see that I got the ball rolling. I can't say for sure of course, but I don't believe that the project would have even started by the time I handed it off if I hadn't gotten frustrated and started it. In the end, I feel I did something good for the company by getting the project rolling when I did.

My goal in all of this is to try and persuade just a few of you perpetual whiners out there to get up off your asses and do something. If you won't do that, try and find someone who appears to be following my system (or something similar) and bitch about your grievances to them. Do it long enough, and they are likely to pull you along with them when they decide to get some damn done.

[1] "Simple" in this case refers to the effort needed to understand the rules of the system. It doesn't mean that either of the steps is necessarily simple to complete.

[2] I didn't speak per se, but my thoughts were eloquently expressed thanks to the Android text-to-speech British chap. Stupid laryngitis.

Erik Davis has more than 13 years of experience in software testing as both a tester and test manager. He discovered his hidden passion for Context-Driven Testing at the CAST 2012 conference.

Since then, Erik has become an active member in the testing community; looking for ways to help people grow and learn as testers.

Erik can be found on twitter at @erikld, on his blog http://www.testingthoughts.com/erikdavis or at the tester meetup he founded, www.meetup.com/NOTiCE

Back To Index


There was a time when people did not have compass to find right direction. The only guide they had was that guiding star up in the sky.
Do you think that you are also stuck somewhere with technical issues? Do you need help in decision making or want guidance?
Well, the wait is now over . Introducing....
"The Guiding Star"
The panel of our experts is now here to help you.
Send us your questions around software testing and our Guiding Stars will help you out.
E-mail your question on –
theguidingstar@teatimewithtesters.com
Please Note :
1. This is not a job portal.
2. Typical interview questions will not be answered.
3. Questions should be on Software Testing or related topics only.
Image courtesy : MMVI New Line Production

International Conference on QA & TEST Embedded Software Testing

12th International Conference on Software QA and Testing on Embedded Systems

This year we offer a special program: **3 Keynotes, 2 Tutorials, 8 Tracks**, from renowned companies. Visit our website -www.qatest.org- to **discover the Programme!**

QA&TEST, the international Conference on Software QA and Testing on Embedded Systems, will be held on 29, 30 and 31 October in Bilbao, Spain, and gives you an excellent opportunity to increase your business network and establish contact with others professionals of the industry.

The main objective of QA&TEST is present the last technological developments in Software Testing and Quality Assurance, and showcase successful best practice to reduce costs and give companies a lead in global competition.

www.qatest.org

SOFTWARE

Organiser

SQS SOFTWARE QUALITY SYSTEMS

Sponsor

innovalia ASSOCIATION

October 29 · 30 · 31

2013

Bilbao · Spain


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

Model Based Testing without Assumptions

- by Silvio Cacace

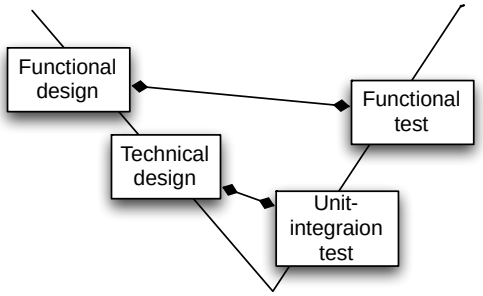
Model Based Testing nicely deals with modern development methodologies. But also in traditional development projects, this way of testing has many advantages over other testing methods. However, this is highly dependent on the way we organize our MBT testing process and what role is played by the tester. Should the tester blindly assume an already established model or should the tester be responsible for the preparation of the test model? In this article, an explanation.

Increasingly, we see that Model-Based Testing (MBT) is used for software testing. This is a logical step to keep our value as a functional tester high. As a tester we simply can no longer carry out testing as we previously did which was accustomed to the traditional development processes. No, especially if we look at the contemporary agile development processes, as a tester we must anticipate the new ways of working. Project properties such as interactive, iterative, incremental and multidisciplinary must therefore be consistent with the current test approach. MBT deals nicely with that, provided that this new way of testing is applied correctly.

The benefits of applying MBT are now known. Thus we can, through the proper use of MBT, address any 'open ends', ambiguities, inconsistencies and errors in the requirements, at a very early stage. This will shorten the duration of the project and improve the quality of the software. Another great advantage of MBT is the flexibility and adaptability of the test set. If changes occur in the requirements, we merely need to adjust the model and then we can generate a new test set automatically. This automatic generation of the test set is another great advantage of MBT. The condition is that we have the proper tool.

With regards to the above, I would emphasize that MBT is the modern way of testing and has many advantages over other testing methods, but this is highly dependent on the way we organize our MBT testing process and what tool we are using.

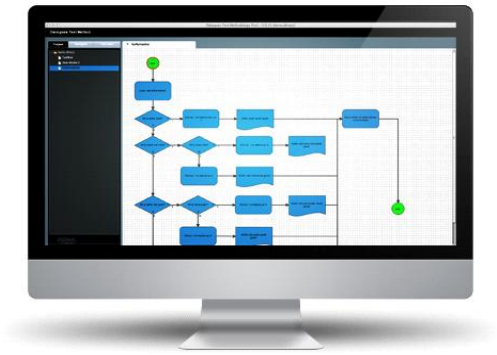
Back To Index



If we look at the traditional development processes, we see that the tester has an independent view on the test basis. Think of the V-model, it shows that the basis for technical testing and development is the technical design and that the tester will use the functional design to set up the test set for the functional test. Precisely this will effectuate an independent view of the tester and any interpretation errors in the requirements will be observed. The tester can make no assumptions and all the required information to establish a thorough test set is collected by the tester himself.

But how does this work in MBT? Is a tester also able to maintain an independent view and does the tester have a clear field to complete the test basis, without making assumptions? The answer is yes, as long as the tester is able to draw the test models himself.

We too often see that within MBT the models that are used to generate the test set, are too technical, static or poorly readable and these models have not been produced by the tester itself. Who can say the author of these models has made no mistakes in the interpretation? Who takes care of updating the models after an update in the requirements? Do all project members, such as the tester, product owner, etc., understand these models? Are all test situations captured in these models? And perhaps more importantly, are these models so arranged that for the generation of the test set, a test algorithm can be used, based on the desired test coverage?



The solution to the above-described problems is simple. In MBT, the functional tester must draw the test models. The tester should then evaluate these models with the team members and business and where necessary adapt the model. Just as the functional tester composes the test set within traditional development processes. With this, the functional tester preserves an important independent perspective, these models are world-readable, these models are easy to adjust by the tester itself and these are really test models. From out of these models, now the test set can be generated automatically based on a test algorithm, taking into account the desired test coverage.

The tool, which will be used, must be adapted to this. The tester itself must be able to draw the test models in the tool and the tool must automatically generate the test set based on different test algorithms (test coverage).

The conclusion is that for functional testing, MBT perfectly fits to modern development processes, but that the value of the functional tester is even greater, with a correct insertion of this method.

Silvio Cacace is working in the software test area since 1993. He has worked as a Test designer, Test navigator, Test coördinator, Test manager and Test consultant for many Dutch companies, as ABN AMRO, ING Bank, University of Amsterdam, Interpolis, Corus, etc. Since 2004 he has been working as a Freelance Test consultant. In these years, he has developed a Test strategy for Agile projects and the Test approach ATS, which is based on TMap®, but practical and pragmatic. The last 3 years he is working for Dialogues Technology and XL team as Agile test consultant. In cooperation with XL team they've developed the DTM tool, which is a model based testing tool (www.dtmtool.com).

Selecting Smart-phones for Compatibility testing

- by Faisal Qureshi

Introduction

As mobile computing continuously evolves, so does the software aligned with it. The market trend for mobile applications has a positive derivative in regards to both players involved and growth. Key operating systems are currently governing the market with various hardware vendors, and thus, developers in the mobile market should account for this diversity. When releasing software for mobile devices, it is critical that the software runs correctly and efficiently on all possible devices. Due to the number of different smart-phones in the market, this makes compatibility testing a difficult task. It is impossible to test the product on every device in the market. Thus, it is imperative to select a set of devices to get the widest possible coverage that lowers the probability of failure on non-tested platforms. To do this, a selection algorithm can be used that uses *Q-distance,* which is introduced here as a new unit of measure for this purpose.

Methodology

Q-distance determines the "distance" from a "seed" to another device. Based on the value, the set of devices to test on can be systematically chosen. The method used to calculate the *Q-distance* utilizes a relative scaling times a priority weight. Multiplying the scale value by the priority weight is performed on a chosen set of attributes of each smart-phone. The priority weight is chosen by the tester which should represent how important the attribute is for the product. After this is done on the chosen set of attributes, the results are summed for that particular device. The result of the sum is called the *Q-Value* (a new unit of measure required for Q-distance calculations). Once all *Q-Values* have been calculated, the "seed" is the device that has a *Q-Value* as a median (or closest to median). This device is called *S*. The *Q-distance* is the difference of *S* and the other devices. Once this is determined, the highest priority set of devices to test on are the ones with the largest *Q-distance*, including *S*, and are categorized as *P0* devices. The *Q-distance* is marked as positive or negative relative to S. At each selection, one device must be from positive, and the other from negative. The next in line are the devices that have the next largest *Q-distance*. This continues till constrained by time and/or resources. This way, there is the

widest test coverage and have reduced the probability of failure on non-tested platforms. If the attribute is not quantifiable, it can be assigned sequential integer values relatively from smallest to largest.

Definitions

Device: D*1* to D*m*

Device attributes: A*1* to A*n*

Weights: W*1* to W*n*

Scale value: S*xy* = (value(D*y*(A*x*)))/(∑value(D*n*(A*x*))) where n=1 to *m*

Q-Value(*y*) = ∑(S*xy* * W*x*) where *x*=1 to *n*

S = Device with median *Q-Value*

Q-distance(*S,y*) = |Q-Value(*S*) - Q-Value(*y*)|

P0 devices = S and other 2 devices with largest Q-distance

P1 devices = 2 devices with second largest Q-distance

P2 devices = 2 devices with third largest Q-distance

Example

Devices:

D1, D2, D3, D4,D5, D6

Attributes:

A1 = dpi, A2 = Memory, A3 = OS version

	A1	A2	A3
D1	96	256 MB	2
D2	126	512 MB	1
D3	200	1024 MB	3
D4	220	256 MB	3
D5	150	512 MB	1
D6	200	512 MB	2

Scale Values:

	S1	S2	S3
D1	0.09	0.08	0.167
D2	0.12	0.167	0.08
D3	0.2	0.33	0.25
D4	0.22	0.08	0.25
D5	0.15	0.167	0.08
D6	0.2	0.167	0.167

Weights: W1 = 3, W2 = 1, W3 = 2

	S1*W1+S2 *W2+S3 *W3
Q-Value(1)	0.684
Q-Value(2)	0.687
Q-Value(3)	1.43
Q-Value(4)	1.24
Q-Value(5)	0.777
Q-Value(6)	1.101

S = D6 (closest to the middle)

Q-distance(S,1)	0.417	-
Q-distance(S,2)	0.414	-
Q-distance(S,3)	0.329	+
Q-distance(S,4)	0.139	+
Q-distance(S,5)	0.324	-

Thus:

P0 devices = D6, D1, D3

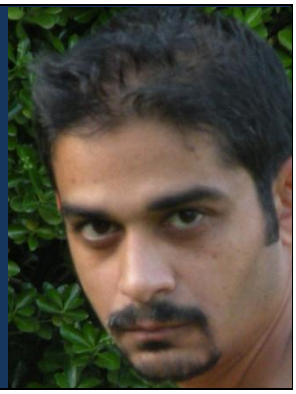
P1 devices = D2, D4

No time to test D5!

Conclusion

By following this method, selecting devices for compatibility testing can be systematically determined rather than making an educated guess. This can be applied to already obtained hardware, or even during a research phase to determine what platforms to acquire. The important concept to remember is that the final Q-distance value is based on important decisions a priori for attribute selection and weight values.

Faisal Qureshi currently works at Amazon as a QA Engineer. Prior to this he worked at Motorola Solutions as a Senior Test Engineer. He graduated from NYU-Polytechnic with a Bachelor's degree in Computer Science and has a Master's degree in Computer Engineering from Columbia University in New York. Faisal also has publications in Professional Tester, Testing Experience, and Better Software Magazine and continues to work on test methodologies and techniques to enhance and further define the science of Test Engineering.



Back To Index


Taking
a break?
a click here
will take you there

Let's Talk (Common Sense)

On 21 August 2013, a new organization, the **International Society for Software Testing** was launched; with a mission to promote an approach to software testing that emphasizes value and the role that skilled testers play in its delivery.

Why?

Testing is a funny old game. Have you ever noticed: many of those we serve have unrealistic expectations of testers. Never miss a bug? Completely test? Automate everything? Most of our stakeholders have never tested, nor have they spent any time studying testing. Why should they? So why should they think differently? They're not testers after all.

What of the testers? Should they not be advising, injecting some realism into the conversation as to what testing can and cannot do? Many do, but many don't know any better. Many are in no position to do anything other than comply with what is demanded of them, or to fail where expectations are impossible to meet. Of course, remarkably few people get up in the morning with the intent of doing a crappy job. There are those who see processes, tools and methods as the solution, as a way to achieve

the impossible. Yet by emphasizing the mechanical, the inhuman, they make matters worse: reinforcing the perception that testing is an unthinking clerical activity best delegated to machines, or failing that low cost labour.

Then we have the vendors, whose marketing machines push the idea of commodity testing, with tests assembled and executed in large scale test factories, and testers driven to hitting daily test case counts. One of us, Iain, once met a project manager who was so focused on the number of tests executed each day that he demanded that testers ask permission to go to the bathroom. In some cases this mind set is not just inhuman: it borders on the inhumane.

What does this mean for the quality of testing? Testing is a process of discovery, and discovery is unpredictable. It is impossible to determine where all of the critical problems lie in advance of testing - but many of the practices associated with commodity testing tell exactly this lie: write the tests, run the tests, measure people on how many tests they are running - and heaven forbid that one of the testers discover something that suggests the estimates were wrong. Further, approaches to testing that emphasize process and methodology over people and communication tend to introduce waste: instead of focusing on identifying and solving testing problems relevant now, on this project, they instead focus on tasks that may or may not be of any value.

Ultimately, testers will fail when they are measured against unrealistic expectations, and testing will fail when it is bent to fit a mould more appropriate to manufacture than to research. We're worried about where this is heading, about a possible future where testers have driven themselves into irrelevance due to an insistence on cookie-cutter practices that add little and cost a great deal. We're worried, in a world increasing dependent on software, about what this means for software quality. As the stakes of software failure increase, it seems to us that testing practice is lagging behind.

It's not all doom and gloom though. Whilst we've seen testing operated this way, we've also seen the alternative: an emphasis on the tester. A tester who is given the freedom to exercise skill and judgment is more able to navigate the software development maze, to understand the objectives specific to the project, to learn what matters, to create, tailor or select practices that are relevant, and to change course when new information becomes available. We know things can be different, and believe that things need to change. Thus the ISST was born, to facilitate such a change.
But how? How do you set about effecting change in an industry? It does, after all, seem like a tall order. The answer lies in economics, in supply and demand.

Supply is pretty obvious: without a supply of skilled people to conduct testing work, then no skilled testing work will take place. The ISST sees the development and growth of a global community of testers as the primary means of developing such a supply. A vibrant community will capture the interest and attention of testers who have a desire to learn, whilst events such as conferences, training, webinars etc. provide a means for testers to share information and ideas. We will encourage and support such activities.

Of course, a supply of skilled testers guarantees nothing: other than self gratification, there is little point in having a skill if no one will hire you to use it. But imagine how different the industry would look if even a handful of large enterprises were to say to their vendors: "Sure, cost is important, but we don't want to spend money on stuff that doesn't add value. Give us testers who are skilled, who can speak our language and who will work with us to figure out what's important". Imagine how it would feel if a large commodity testing vendor were to declare, "OK, this doesn't really work, we're going to do something different". Imagine if a significant number of hiring managers were to consider the ability of prospective

testers to think and communicate, rather than their ability to repeat methodological buzzwords. Such a change cannot take place solely within the confines of the testing community. Such a change must take place in the minds of those who make decisions: about how to source testing, about who to hire, about how testing is viewed on a given project. Therefore, the ISST will pursue an advocacy agenda and seek to engage with executives, project managers and developers in order to raise awareness of the issues of skill and value, and to encourage a change in mindset.

Our goals are ambitious and we need your help. Indeed, it would be arrogant in the extreme to assume that we have the monopoly on ideas and energy: we are actively seeking participation from volunteers who can help us to fulfill our mission. For more information about the International Society of Software Testing, and to get involved, please see **http://www.commonsensetesting.org/** and follow us on Twitter: @IntSST.

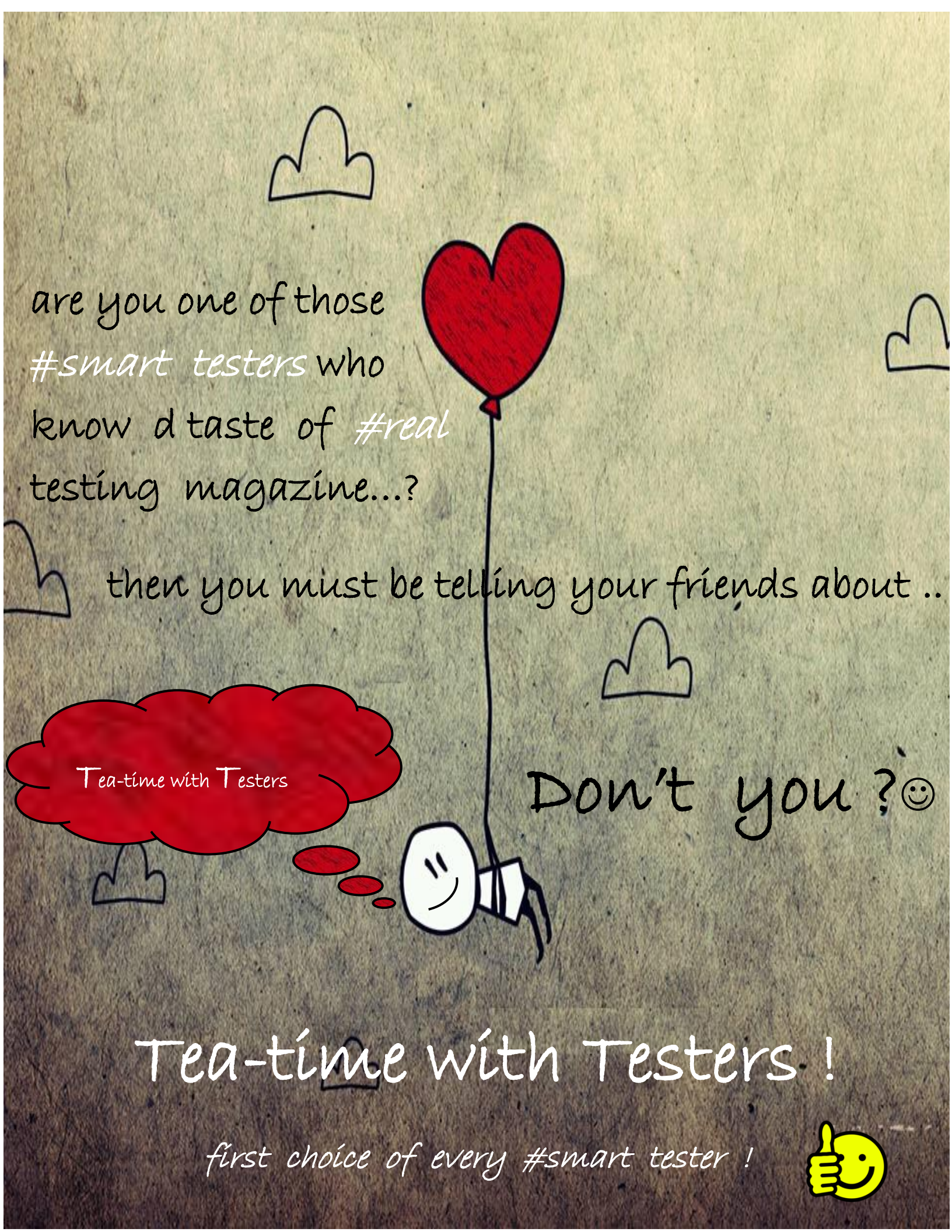
Ilari Henrik Aegerter leads the Quality Engineering Europe group at the world's biggest online marketplace eBay where he is supported by magnificent test professionals. Quite some time ago he became a software tester by pure chance because he urgently needed a job during his studies in general linguistics. He then so much liked the profession that he continued to intensively work on his skills. Today he is an avid follower of the context-driven school of software testing and he believes that software testing is not a clerical job but a profession that needs a high level of proficiency.

He believes that people are generally good and that there is plenty for everybody in this world. All that results in him smiling a lot. @ilarihenrik

Iain McCowatt is a tester, test manager and occasional automator whose experience and passion for testing spans more than a decade. Despite having worked on a wide variety of projects across multiple industries, he continues to be amazed by the degree to which testing is misunderstood; by testers and their clients alike. He is currently a director at Barclays where he is fortunate enough to work with some of the best testers in the world. In his spare time Iain is active in the software testing community, serves as an instructor to future testers, and blogs at exploringuncertainty.com @imccowatt

Johan Jonasson has worked as a test consultant and coach in Sweden since 2006. He's one of the co-founders of the consultant company House of Test, as well as one of the originators behind the Let's Test conferences. The context-driven testing community has been Johan's primary source of professional inspiration since 2007 and helping that community grow is one of his main professional goals, which he hopes ISST will be play an instrumental part in achieving. @johanjonasson

Henrik Andersson is Co-founder and CEO of House of Test Consulting. A consultancy based in Sweden, Denmark and China focused on context driven testing. To co-found ISST is a natural evolution for Mr. Andersson to help strengthen, evolve and grow the Context Driven Testing community. @henkeandersson


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 !

He really didn’t know much about testing until somebody offered him a job of a tester.

That was back during first bubble, people didn't really know a lot about testing and there were not a lot of resources to learn from. He bought some testing books and started his study…

Today, he is author of top-listed software testing blog, a book and has also started PractiTest.

We spoke with **Joel Monetvelisky** this time; to discover a lot more interesting things that you would love to know about……

Over a Cup of Tea with Joel Montvelisky

We would like to start with your journey in testing. How do you feel about your journey from Joel Montvelisky to 'The Joel Montvelisky?' ☺

I am not sure who "The Joel Montvelisky" is. When I look in the mirror I see the same person I have been all my life, special in some ways and normal in most of the others, just like everyone else. I don't think I do anything special other than speaking my mind openly, even when people may not really want to hear what I have to say ☺.

I will try to answer about my personal journey into the world of testing, although I really think that it is not something special or exciting...

I really didn't know anything about testing until someone offered me a job as a tester when I was back in the University. A friend of a friend came to visit and asked me if I knew someone who would want to do some testing in his new Start-Up. As I didn't have a job back then I said "sure" and I started working as the first tester and employee number 7 in an internet startup.

Within a couple of months we were 3 in the team and as I was the first one I was named the Test Lead. That was back during first bubble, people didn't really know a lot about testing and there were not a lot of resources to learn from.

I bought a couple of books and that's how I started to learn about testing. The company grew fast and so did my team. By the end of the first year I had about 15 testers divided into 2 testing teams. I stayed with this company for about 3 more years, but then they relocated to the Silicone Valley and I chose to stay in Israel. I went to work with another startup for a year, until they also relocated to other place and that's when I chose to look for a larger company that would not relocate soon...

Within a couple of weeks I got one cool job any tester would aspire for, I became the QA Manager for Test Director in Mercury Interactive. One of the cool things there was that; we didn't have license issues with any of the testing products :-) But to tell the truth, other than that it was just like any other QA job...

After a couple of years I moved to manage the QA for other Mercury products. I also started an initiative called QA-PSO where I was in charge of interacting with the Professional Services organisations for Mercury worldwide and I would send testers from our QA to help PSO engineers on their harder cases with customers, where they needed really advance knowledge of the tools and the technologies behind them. What we found was that testers had an almost natural ability to interact correctly with customers, and at the same time we could learn a lot from the way they worked with our products and we could take this information back into the testing and the development teams as feedback.

I stayed in Mercury overall for 5 years, and when they were bought by HP I decided it was time to move on.

After Mercury I worked as a testing and QA consultant for about 2 years. This opportunity helped me to learn a lot about different testing teams and processes and made me aware of a serious problem, there were no tools that could fill the gap between Quality Center on the one hand and excel + bugzilla on the other hand. And this is where the idea behind PractiTest came along.

Today, about 5 years after starting PractiTest, I work mostly as a solution architect. I interact all the time with PractiTest customers, learning about their challenges and the way they work and trying to help them to solve their issues with the proper methodology and with the correct usage of their testing tools. I also manage the internal testing process in PractiTest and I try to be as involved as possible in the design of the product and the vision of the company.

QA Intelligence is the word we heard first from you. What made you coin it or choose it as your blog name?

QA Intelligence is the best way to describe our work as testers.

In a nutshell, I believe the objective of testing is to provide actionable information and visibility to all the project stakeholders, allowing them to make decisions related to their work.

This information is similar to the one provided by Intelligence Units in most modern armies.

Just like Intelligence Officers, as testers we need to work with partial information, trying to understand the true status of the situation as we "walk around it", gathering information from multiple sources, and working hard to paint a complete picture under all these complex and changing circumstances; hence QA Intelligence.

On your blog, you have written on almost every aspect of software testing. How do you manage to write from your busy schedule? And more importantly what secrete is there behind your knowledge about various dimensions of testing field? Any advice for our readers?

I wish I could write as much as I want, but it is hard to find the time to do it right. I keep notebook with me at all times. I use it to write a lot of stuff. Among the things that I write are ideas for possible blog posts.

Sometimes I take notes during meetings, other times I do it when I am reading an article. It is interesting that most of my blog ideas come from work sessions and conversations I have with testers as part of my work in PractiTest. It is really important to write down your ideas with as many details as you can. Write them down as soon as you have them and don't worry about the form at this stage only about the content. Later on you will have time to complete your writing and polish your work.

I believe that good posts come from the writer's heart. The writer doesn't need to be an "expert" on the subject, but he or she needs to have a real and personal "experience" and spend some time researching the topic to make sure they understand it clearly. I do most of my writing in the morning. This is the time when I am in "writing zone". I realised that if I write something after 2 PM I will end up modifying it completely the next morning. It is important to find the time of the day and the correct atmosphere that will help you write.

So for me, whenever I have time (usually once or twice a month) I will start the day early in the morning in one of the coffee shops next to PractiTest's office and sit down to write a post based on some notes that I took during the previous days.

What are the 5 key characteristics of 'Professional Tester' according to you?

I think that most of the characteristics that make a Professional Tester are similar to those of many other professionals in the work fields related to software, hardware and technology in general. It is hard to come up with only 5, but I will try:

Natural curiosity - As a tester you need to be curious and have an internal hunger to understand the secret of all things around you. You need this to learn new tools and methodologies, as well as to keep testing and working with applications that you've already tested a number of times in the past.

Unless you approach your job as a learning experience (maybe even a "learning experiment") you will not be able to be a good tester for long.

Communication skills – Every tester needs to be an expert communicator.

I'm talking about communication in the sense of knowing what to say, to whom, when and how, in order to provide the information required by each of our stakeholders in a correct and timely manner.

I am also referring to communication in the sense of learning how to get information from all possible sources that will allow you to do your job better. This information can be as specific as what was changed in the product that needs to be tested. But also as vague as what are the main business challenges in your project in order to get a better idea of how to define your testing and product risks map.

Technical cleverness - A good tester needs to have a technical sense that will allow him to perform his job efficiently.

Technical cleverness is an adaptation of Janet Gregory's "technical awareness". But I think that as testers we need to be more than aware technically, we need to have an active technical side that will let us get into the more complex aspects of our project. This will help us read the changes in the code and make up our minds on what needs to be tested (as a compliment for the information we can get from our development peers), and it will also allow us to develop or work with the technical tools that will help us perform our jobs more effectively and efficiently. A tester does not need to be a developer, but he cannot be afraid of submerging himself/herself in the technical waters either.

Thirst for self knowledge – A tester needs to develop the tools that will allow him to learn by himself and not only from the frontal lessons he gets from teachers or mentors.

Once I thought this was only my personal experience, now I know that this is more common than we all think it is:

As a junior tester I had no one I could I learn from. On the one hand all the testers in my team were inexperienced just like me, and on the other hand my manager came from the field of software development and he was not really skilled or even interested in understanding more about testing. So I took upon myself to learn from as many sources as possible, books, sites, and even other testers I met in professional and social interactions. I was not passive about my thirst for knowledge, and I made this a priority in most of my activities and interactions. Back in those days we did not have Blogs or Web-based testing communities, so it was very old-fashion learning, but it did the trick and helped me to become a better tester day by day.

I think that today junior testers have an easier time finding knowledge sources that are both free and accessible. So it should be easier to be a self learner and to keep expanding your knowledge in testing.

Cynical optimism – this is something that I think can only be understood by another tester: how can you be an optimist and cynical at the same time??? Well, when you are testers, this is exactly what you need to be in order to test and not go mad as part of your job.

Optimism is what allows us to start each testing project thinking that our development peers are working in the most professional way, trying to fulfill their tasks without doing a sloppy job and inserting unnecessary defects into their code. But at the same time cynicism allows us to look into each feature and screen of our products anticipating every single defect possible in order to catch it, reproduce it, and report it.

What is your opinion about test metrics? Evil or Important?

Metrics are tools. They cannot be good or evil by themselves. Only testers or managers who use metrics in the wrong way can cause harm and turn them into evil mechanisms.

In a sense metrics are like a knife. A thief in the street may use a knife to hurt you while trying to steal your wallet, and then you will get to the hospital where a surgeon who will use another knife to save your life…

Disregarding metrics and working only with gut feelings is not only wrong, it is also dumb. But looking at metrics without interpretation or without understanding the context of the project is even dumber sometimes evil. We should use metrics wisely, to help us understand the reality of our projects but not to serve as the only channels or methods of information that will help us make up our minds.

How much important is 'hands on testing' for test managers and leads, according to you? Especially in case of service industry, leads and managers are mostly occupied with billing and other activities. What would be your advice if they want to get started? How can they find balance between things?

A manager NEVER has enough time, this is a fact.

As soon as you realize there's not enough time you start prioritizing between your tasks, setting aside the time to perform those tasks that are imperative to make your job right. Understanding the work of your team is one of the things you need to make your management job right. Understanding the application your team is testing is another one of those things.

If you have other ways to REALLY understand the application and the job of your team, then by all means feel free to use this other way. If you don't, then don't make excuses for it.

For me, the best way to understand the job of my team and at the same time to get a feeling of the AUT (Application Under Test) is by taking part in some of the regular testing tasks.

You don't need to run 5 straight days of testing, but you can set aside 2 or 3 sessions of 45 minutes each to do pair testing with some of your testers. This way you get to test the system and feel the work of your team, and at the same time you save the time it would take you to learn your AUT and master the testing scripts by yourself.

If you don't have time even for this, why not taking 30 minutes to go over a guided review of the critical bugs reported by your team during the last couple of days? This is a less hands-on approach, but it will still give you some feeling of the AUT and the main tasks your team is running.

Is automation testing in high demand in industry? What is your opinion about test automation?

Automation is a tool, just like Exploratory Testing, Mind Maps, etc.

Every tester carries with him a "virtual toolbox" where he is constantly accumulating new tools for his testing activities. Automation and scripting is one of these tools, and a pretty good tool to use in many occasions.

On the other hand, just like any other tool, there are some people who take the time to develop an expertise on automation, while others only know how to work with it in order to solve small problems. Both approaches are OK as long as you know what you can and can't do. I am sure, most of us don't have the skills it takes to build a house, but we do have enough experience with a hammer to hang some pictures on the wall.

Same way, when you want to use automation extensively in a project it makes sense to have an automation expert in team to do the work, and not to rely on someone whose automation knowledge is limited.

What will be your advice for job seekers who want to make career in software testing?

My advice to the job seekers looking to start a career in testing: start working NOW, make no excuses.

When I talk to someone looking for his first job as a tester I explain to him/her that it is a bad excuse to say that they have not done any actual testing because they have not found a job as a tester in a company.

There are crowd sourcing sites, for example uTest, where you can start getting your hands dirty in testing tasks, gain some experience, learn valuable lessons, and even get some money at the same time; all without the need of experience and even while working and studying part time.

In parallel, don't waste any more time, go out and look for a job, even if it is without any pay at first, so that you can get some experience. Call all your friends and tell them that you are willing to work for free for a couple of months in exchange for a job in the future if you are good enough, or even for a references in your CV.

If you are technically sound (or if you are not afraid of learning how to become technical tester) download a free testing framework such as Selenium or Watir and learn how to automate windows applications or regular websites. This is also a way of gaining experience!

In short, act now, start practising and don't wait until you are made an offer.

Which software testing books do you like? Please tell us about your free eBook 'The Road to QA Excellence'.

My favorite testing book is **Agile Testing, by Lisa Crispin** and **Janette Gregory**. I think that the title can be a little misleading at times, because the advice they provide in their book can be applied to any type of testing process and not only to Agile development and testing.

There are other books that I like to review once in a while, one that comes to mind is **Lessons Learned in Software Testing by Bach, Kaner** and **Pettichord.** It was one of the first books I remember reading and making a difference to the way I approach my testing processes.

The Road to QA Excellence is an eBook we created at PractiTest with a collection of thoughts that can help any tester, at any stage of his career, to make sure he is at the top.

It has 7 steps (6 steps and a bonus step) that can help even an experienced tester to make sure he is at his best and to ensure he is constantly improving the way he works and the value he provides to his organisation.

A nice thing about this eBook is that we made it based on both our expertise but also based on endless conversations and interactions we had with some top-notch Test Managers who work with PractiTest.

You can get it from free from our site, and we'll always be happy to get your feedback on it.

You have rich experience of working and designing Test Management tools. Do you think that test management tools have influenced the way things are measured and processes followed in testing field (like pass/fail metric, % execution metrics, defect related metrics)? Or it's vice versa?

I would say it is vice-versa, and I hope that other tool developers listen to their customers and users with the same attention as we do in PractiTest.

An important number of the features we developed in PractiTest came from talking to customers and understanding how could our tool be expanded and improved in order to better serve their needs.

With the years, and as PractiTest has become more mature we get less requests to improve existing features and more requests to continue expanding the system into other areas of the testing and development process. All these are request that we are carefully examining, and that we will continue to review in order to keep developing PractiTest always based on our users needs.

Is there any story behind PractiTest? We would like to know.

There is always a story behind everything ☺

PractiTest is the result of 3 friends who got together after we understood there was no tool that could provide a good (methodological) solution for companies that didn't want to work with large enterprise solutions such as QC or TFS, but that were looking for something more professional than the Excel sheets to manage their testing.

The 3 founders had worked together back in the late 90's during the first bubble, and we kept in touch on and off throughout the years. Then, after I had been working as a QA Management consultant for close to 2 years we got together and started working on the PT concept.

One of the most interesting facts about PractiTest is that up to now we have not yet needed to get external investment for the company. We started while working on other projects and soon enough, as more and more companies stated working with PractiTest, we found that we didn't need to get external capital in order to build the company and keep growing.

I guess this means that from the beginning we created the right product, and that it answered the needs of our users and the field.

How does it feel being part of Tea-time with Testers family? Any message for our readers?

For me, it is an honor to be considered as part of the Tea-Time with Testers family.

Team is doing an incredible job of putting together such a great magazine with such a high level of articles every single month.

My message to all the readers is to keep sharing the magazine with their peers and to actively contribute to it.

One of the biggest assets any magazine has, are its readers, and specially those who get involved and contribute in the form of articles, comments and simply referring it to other testers worldwide.

Team, keep up the good work! Dear readers, thanks for contributing and making this magazine successful!


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

Not more, but no more.

When we want to do significantly better, the tendency is think that we need to do more. More tests, more cycles, more automation etc. It is only natural to think that we need to accomplish more. This puts a lot of pressure, and the challenge is to ensure that we respond to this well, not buckle. In the next few paragraphs, I am going to argue that it is not about 'doing more', but it is really about intense focus to 'do no more'. That is, remove fluff to 'do no more'.

Seems contrarian eh ...

Let me illustrate this with a story. When I got into long distance endurance cycling where the minimum distance of a brevet is 200 KMS, I thought it is about building stamina, fitness and got to doing more - More rides, more time on the saddle, more speed, more distance. The challenge I faced was that it was becoming difficult to ride for a longer time as I was constantly looking at the clock and the milestone marker, sadly both seemed to move slowly and it was frustrating. That is when I decided to do something different - to focus only on the front tyre and ride. What this did to me was amazing. It allowed me to see only short distance in the front and not worry about all the 'mores' i.e. distances, time, speed. Suddenly I was in the present and could achieve higher saddle time by 'doing no more'.

The trick was to sharpen the focus; in this case this was the distance to look at and doing just what you need do, no more.

When you want to find 'more' defects, focus on the objective of what types of defects you want to uncover. Focus on examining information that can give you a better clue to the types of defects. This goal focus enables to come with just enough (no more) and not necessarily more test cases.

When we want to deliver higher quality software, it is not about doing more testing. It is about focusing on the types of defects that could creep during the entire SDLC, and then figuring how to detect earlier or prevent and therefore 'do no more' rather than 'do more'.

When an application has matured and is in the maintenance phase, number of defects uncovered by the (typically) large set of test cases is low. Here again it is not executing more, it is focusing on which test cases have become 'immune', and then focusing on those that have higher 'potency' based in the potential defect types that could be irritated by changes done. Here again this is another case of 'do no more'.

When we want to shorten release times, it is not only about more automation to execute more to shorter time. It is also about focusing on those types of defects that are more probable and executing those i.e. 'do no more than necessary'.

'Doing no more' requires us to focus, to think and analyze deeply. It is about using the brain, not the brawn of (more) hard work. And this is the fun part.

The pressure situations, where the objective is to accomplish more is wonderful. Because this is when we can focus on 'what to remove' and 'do no more' rather that 'physically work hard' to 'do more'. Personally I love these situations, as they help me focus sharply and drop all the fluff.

So when somebody tells you to do more, correct them so that they understand this as 'no more'. Pressure situations demand brevity (do no more) and this is possible only when you focus, be in the present, think well so that you can do only those that matter.

The doctors in the emergency room face intense pressure. They need to make split second decisions and respond quickly. Brevity is the key here. Saving a life is not about doing more, it is about just doing what is required -'No more'.

Well, I better shut up.

No more.

May you accomplish more.

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

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

He can be reached at **ash@stagsoftware.com**

Back To Index

by

Testing PUZZLES by Sebi

Claim your Smart Tester of The Month Award. Send us your answer for Puzzle and Crossword b4 20th September 2013 & grab your Title.

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

Puzzle

Find as many as possible sub-directories in https://www.paypal.com/**webapps**/

Example:

https://www.paypal.com/**webapps**/mpp/

https://www.paypal.com/**webapps**/helpcenter/

https://www.paypal.com/**webapps**/cobapp/

Tips: use Google, scanners, exploratory testing

Back To Index

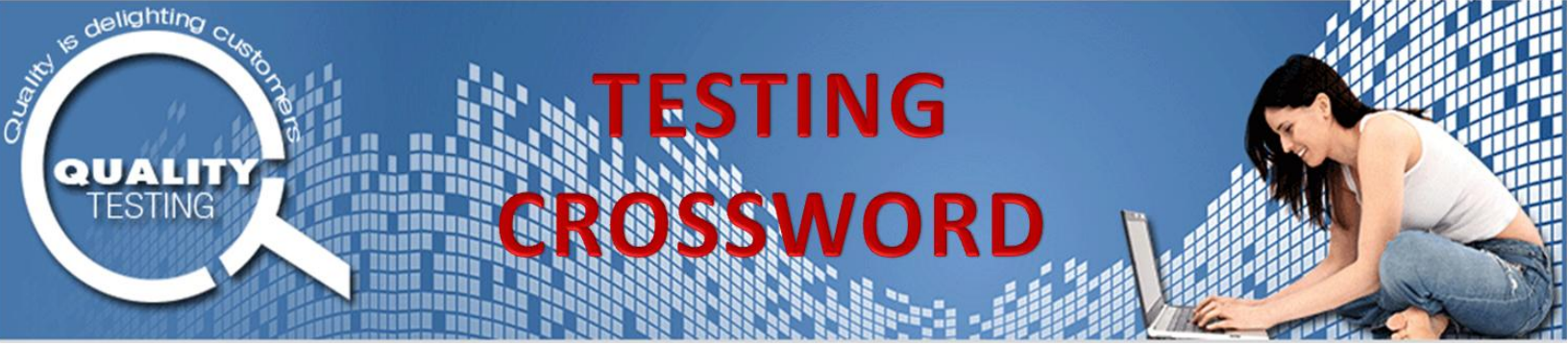
Biography

Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years.

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



1		2		3		4
	■		■		■	
5		6		7		
	■		■		■	
8		9			■	
■	■		■	10		11
12		13			■	

Horizontal:

1. Twitter launches a Beta testing program on_____ (7)

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

6. It is the combination of black and white box testing (4)

8. Checks for memory leaks or other problems that may occur with prolonged execution. It is known as _____, in short form (2)

9. It is one of the most common software testing strategy used in software development, in short form (2)

10. Statistical testing using a model of system operations (short duration tasks) and their probability of typical use, in short form (3)

12. A white box test design technique in which test cases are designed to execute decision outcomes, in short form (2)

13. It is the process of putting demand on a system or device and measuring its response, in short form (2)

Vertical:

1. _____methodology is an alternative to traditional project management, typically used in software development (5)

2. It is a GUI test tool and automation framework written in Python (7)

3. It is a Command-line tool for generating data for Random Testing and Field Testing (2)

4. The variation in the expected and actual results is known as____ (6)

7. Testing phase where the tester tries to 'break' the system by randomly trying the system's functionality, it is known as _____ testing (5)

11. Commonly used to refer to the automated test procedure used with a test harness. It is known as ____, in short form (2)


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

in next issue

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)

Lalitkumar

Pratikkumar

Contribution and Guidance:

Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)

Jerry

T Ashok

Joel

Editorial| Magazine Design |Logo Design |Web Design:

Lalitkumar Bhamare

Core Team:

Dr.Meeta Prakash (Bangalore, India)

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