THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

FEBRUARY 2012 | YEAR 2 ISSUE I

Anniversary special

James Bach in Guest Editorial and Interview

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

Can Testing Be Killed?

- James Bach

Although Rome wasn't built in a day, it took six days to accidentally burn down in 64AD. It was rebuilt to be a bit more fireproof. And with the burning of the Iroquois Theater in Chicago killed 602 people in 1903, the fire code for theaters improved in 1904. The Triangle Coat Factory fire in New York City (146 dead) led to the founding of the New York City Bureau of Fire Protection, and the National Fire Protection Association now maintains several hundred separate codes-- many of them inspired directly by specific tragic fires. People learn from disasters.

This is also why we have testers. Software disasters happened and people learned. Those specific people became more careful, dedicated more energy to quality assurance (including testing), and there were fewer disasters. But, unlike fire codes, devotion to QA is generally not a matter of law. If an organization hasn't had a disaster in a while, their practices get steadily riskier (partly because younger and more innocent people replace the experienced ones). This is a normal Darwinian cycle.

So it’s not entirely surprising that at the STARWest conference, in 2011, Google’s James Whittaker announced that testing is "dead." What? Testing is dead?! He seemed to be saying that testers are no longer needed in a world with automated checks and automatic updates. But Whittaker is not now and has never been a professional tester. Imagine a cabinet factory industrialist, never himself having built a cabinet, announcing the death of skilled carpentry. That’s what it sounded like to me.

As if the Fates had overheard him and been offended, a few weeks later a bunch of Google bugs made news: an article appeared on CNN.com with the lamentable title "The week Google really messed up." A couple months after that, Google Wallet was discovered to have a serious security problem affecting all users. More bad publicity.

What does that mean for the testing field? After all, no testing process is guaranteed to save us from all bugs. Meanwhile, other processes can find bugs, too, or prevent them. Amateurs and part-timers can find bugs. Programmers can test their own code. Just because Google gets embarrassed now and then by bad software doesn't automatically mean they should hire more testers. Maybe instead they should hire better programmers, or train them better.

Well, one thing is obvious: bragging about how you don't test hurts you later when you are begging for forgiveness from your customers. Eh, Google? Setting that aside, what it means for testing comes down to the beliefs people have about the nature of software products, users, software development, economic factors, and the nature of testing itself. And don't forget the ethical and legal landscape. A lot of factors are involved, here.

What Would Kill Testing?

Testing is not dead. Testing won't be dead. And anywhere testing seems to die it will be reborn, phoenix-like, from the consequences of its own ashes. Still, it can be a good exercise to think about what might cause the death of testing, even in a temporary way. Michael Bolton and I sat down recently to brainstorm on that. Here's what we came up with:

1. Testing may die if you start using the word "testing" to mean checking. One of Michael's recent contributions to the craft was to suggest a sharp distinction between testing and checking. To test is to question the product so as the evaluate it. Testing is an open-ended investigation that cannot be automated. To check, however, is to gather specific information and analyze it in a manner that could, in principle, be automated. In the parlance of philosophers, checking is a mimeomorphic activity; testing is polymorphic. Some people, mainly programmers who don't study testing much, believe strongly in automated "test tools." In pursuing their vision of applying these tools to testing, they inadvertently dumb testing down. They do with tools what tools can do. They run many checks. Testing for them becomes little more than a command-line switch on the compiler ("-t for test", or -q for "put the quality in"). And such checks are capable of finding bugs, just not nearly the breadth and depth and variety of bugs that a skilled human can, especially if that human ALSO uses tools.

Mistaking testing for checking can kill testing, in a sense, by co-opting testing practice. Testing, as Michael and I see it, would still exist, of course. But it would be relegated to the shadows. No one would systematically learn how to test, anymore.

2. Testing may die if the value of products becomes irrelevant. It dies when we don't care about the quality of software or the people who need it. By the same token, if we always trusted the water we drank, or the meat we bought at the store, then water testing and food hygiene standards would be irrelevant.

There really is a problem in our industry with the erosion of the expectation that anything will work reliably, ever. I was trapped outside my house, in the cold, recently, and found that my Android phone would not make any calls on the cell network. I rebooted the phone (that takes a few minutes). Still no joy. I connected to wifi and tried to call that way but got a strange error about not being registered. I had made calls through wifi before from my house, so I knew it could work. Finally I started Skype and IM'd my son (this was through wifi, so why didn't the phone calls work?). I'm annoyed by my phone, but not surprised.

Google probably thinks I'm not going to give up my phone just because of a few glitches. What they need to understand is that this creates an opportunity for competitors to come in with a better product that kicks them out of the market.

3. Testing may die if the quality of testing work is chronically poor. Unfortunately, the death of testing can be a self-fulfilling prophecy. People most likely to believe that testing is dead are – like the folks at Google —unlikely to devote themselves to the study of it. They simply don’t know how to test, or perhaps don’t care. It’s only a matter of time before management wonders why they have testers at all.

The antidote for that is a high standard of personal excellence. This is what the Context-Driven testing community stands for. We are doing our best to win over the rest of the testing world by being good role models.

4. Testing may die if all the users in the world were early adopter technocrats. Let’s pretend that all the people in the world who use computers or rely on them in some way are highly technical and tolerant of problems in the products they use. Then the need for testing would dramatically fall. Sure, they want great quality, but if they don’t get it, they understand. For minor glitches, they will have the patience to find a work around. That may be more true for the perpetual beta products that Google famously offers, but everyone on Earth depends on computers in some way, even if they’ve never seen one. And a tiny minority of those people will experimentally download a tool like Google Earth, as I have, and then spend an hour re-configuring it so that it will actually run.

5. Testing may die by suffocation. If testers are forced to channel all their ideas through a limiting set of artifacts or tools, their productivity may collapse. I'm talking about elaborate test plan templates, test script templates and test management tools; and Cucumber "executable specifications" or other automation tools that require the tester to express himself only in stilted and limited ways.

That will kill testing because it turns testers into tool jockeys, whose standard of success is the weight of paper or volume of data or lines of code – none of which has much to do with testing. Tools can be marvelously helpful in moderation, but the excellent tester will resist obsessions with tools, documents, or anything that systematically impedes the variety and profundity of his work.

6. Testing may die if technology stops changing. Testing is questioning the product. There isn't much call to question a product that stays the same, especially if it operates in an environment and for a user base that also doesn't change. The ambition to innovate is what invigorates the need for testers. Take away that ambition and we all will have to get jobs in comic book stores.

7. Testing may die by starvation. When companies reward people who take unknown risks, but not people who discover what those risks actually are, testing begins to starve. If the craft becomes uninviting to smart, talented, motivated people because you've turned it into a boring, uninteresting activity: that also will starve testing. The only people left would be the ones who are too frightened or lazy to leave. The reputation of testing would become steadily worse. The word "tester" would come to mean a non-technical, uncreative, unhappy bean counter.

Michael and I teach Rapid Software Testing, which is like a martial art of testing. It's exciting. We are trying to show people that their jobs don't have to suck. We are trying to feed the testers. By the way, we are booked months in advance for our testing classes. Apparently, if testing is dead, nobody has told our clients. Do you think buggy whip salesmen and corset factories are doing this well?

Final thought: *I wonder when testing ever was truly alive to people who wish to have it declared dead?*

My special thanks to *Michael Bolton* for helping me in this editorial.

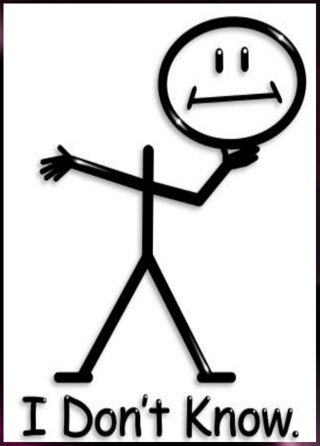
I also congratulate *Tea-time with Testers* for completing one year and hope that you'll continue liking it.

Enjoy Reading !

[illegible]

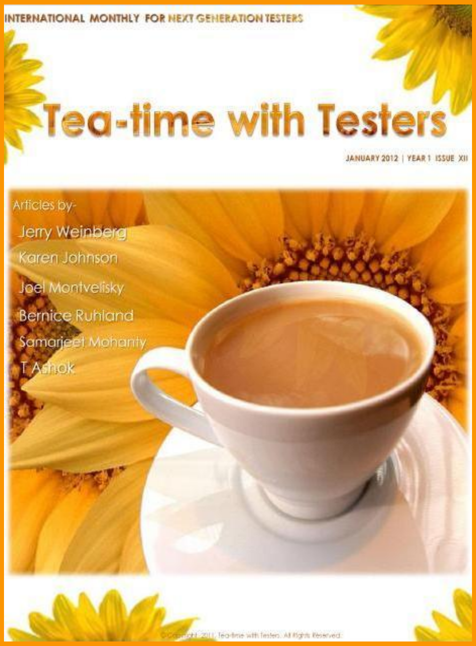
- James Bach

QuickLook



Testing Puzzles
by Sebi

Crossword by



January 2012

Congratulations for completing a year.

I did of course like writing for *Tea-time with Testers*. One thing I like about Tea-time is the passion the editorial team has shown towards Software Testing.

It's not an easy job by any stretch of imagination to bring about monthly issues and that too made so comprehensively well. All this is not possible with a pure passion at the core. Congratulations to *Tea-time with Testers* team for completing a year.

I would just say maintain continuity and quality. It's always hard to surpass the high standards when they are set, so please continue to strive to reach greater heights.

- Anuj Magazine

May this be the first of many happy returns of the occasion.

Many of us have thought about publishing a magazine, but very few follow through. Of those who follow through, 95% publish a few issues and then run out of ideas, so when I saw the first issue of *Tea-Time with Testers*, I was skeptical. Not that there aren't enough worthy ideas about testing. Quite the contrary. But most would-be publishers simply lack the breadth of knowledge to cover the testing profession from all angles. And even if they do, most lack the imagination to publish those ideas in a useful and visually appealing format.

Well, it's now been a whole year full of monthly issues, and *Tea-Time with Testers* has proved itself to be the one exception--so much so that I look forward to reading it every month, and am proud to be associated with it and its marvelous crew of writers and artists.

So, congratulations on not just surviving, but thriving, through your first year.

May this be the first of many happy returns of the occasion.

- Jerry Weinberg

Remarkable achievement

It was a great experience writing for Tea Time with testers.

The magazine provides a good balance of educational content, opinions and testing tool coverage.

What makes Teatime with Testers unique is that a magazine published in India, is now read around the world in 90 different countries. This is a remarkable achievement in a short period of one year.

-Karthik S

It was very enjoyable to write for *Tea-time with Testers.*

I like the community-driven nature and that the articles actually come from practitioners.

- Markus Gartner

It's hard to think it's only been a year.

I really enjoy writing on software testing and seeing it published in *Tea-time with Testers* only makes it better.

I appreciate the fact that you are able to mix some of the biggest names in QA and Testing together with newbie and so show many faces of the testing world. You also have managed to cast a global writing team, pretty impressive.

Overall I think that many people were Skeptical about the need for "another testing magazine", but the fresh look and the depth of the articles in *Tea-time with Testers* has actually proven that you can bring more value than many other publications with tons more investment.

It's hard to think it's only been a year.

I wish all of you in the magazine lots of additional publications and that you keep working with the same level of enthusiasm and professionalism you've done up to now!

I am honored to having helped (in my small way) with your efforts.

Keep it up and congratulations on a great first year!

- Joel Montvelisky

I love being part of such reputable community.

For some time, I was looking for an online community where I could share my knowledge and at the same time learn from others valuable experiences. *Tea-time with Testers* was the place and it's been an honor to write for such a reputable community.

The vast experience and passion of contributors is one of the things that drove me in being a part of this community. I love the funzone in the issues too.

Congratulations for completing one year !

- Samarjeet Mohanthi

Freedom of writing

I enjoyed writing for Tea-time because it allowed me to shares ideas and thoughts on an atypical topic with many readers.

I think the Testing profession is very diverse and feel Tea-time will help us all explore it.

I appreciate the electronic format, monthly frequency, articles from many Testers, and articles from Bach and Weinberg.

In general, I look forward to my Tea- Time with Testers every month.

- J DeMeyer

TO SEND YOUR FEEDBACK; WRITE TO US AT

– editor@teatimewithtesters.com

Teen finds bugs in Google, Facebook, Apple, Microsoft code

Elinor Mills , February 2, 2012

When he's not at school, 15-year-old Cim Stordal spends his time playing the Team Fortress video game, shooting his Airsoft pellet gun, and working in a fish shop in Bergen, Norway. But his real passion is finding bugs in software used by millions of people on the Internet.

Stordal has made the Google Security Hall of Fame, been credited with disclosing a cross-site scripting bug to Apple, been thanked by Microsoft for disclosing a vulnerability to the company, and received an elite White Hat Visa card from Facebook with $500 credit on it.

"I got a card for a self-persistent XSS [cross-site scripting flaw] at Facebook, and a nonpersistent XSS at Google, Microsoft, and Apple," he said in a recent Skype interview with CNET. (As a "self-persistent" issue, the bug Stordal disclosed was not exploitable by a third-party because it required a user to take an action to be at risk, according to Facebook.)

"I just look around at the site and find out where I can input HTML and stuff and it's not filtered in the source code. Often they filter some characters but forget some or they totally forget that input," he said. "What an attacker wants is often the cookie, which can be used to log-in as the user."

Stordal says of the sites he poked around in, Apple was the easiest to find a flaw in. "I found the Facebook [hole] after four days and the Google one after three, but Apple took me only five minutes" to find two XSS flaws, he said. (Apple representatives did not respond to a request seeking comment.)

The companies appreciate his efforts, particularly because he tells them before going public with any of the details. "Everyone was happy about it and fixed the flaws kind of fast."

Stordal started looking for vulnerabilities in software when he was 14 years old. "I have always loved being on the PC and I already was programming some C++," he said. "So I wanted to do something new and I searched around and learned Basic."

His friends are impressed with his skills and lean on him to help keep their Web sites secure. His parents aren't really sure what to make of his research.

"They think it's kind of cool, I guess, as they don't understand what I do," he said. "But they also don't want me to stay on the computer all day."

His next move is to look for vulnerabilities on mobile devices. He's trying to set up a fuzzer (automated software testing tool) on his iPhone 3GS.

About Elinor Mills

Elinor Mills covers Internet security and privacy.

She joined CNET News in 2005 after working as a foreign correspondent for Reuters in Portugal and writing for The Industry Standard, the IDG News Service, and the Associated Press.

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

Are you interested in publishing the news about your own firm, community, conference etc in Tea-time with Testers?

Feel free to write to us at: contact@teatimewithtesters.com with "News Enquiry" in your subject line.

Back To Index

Announcing...

Smart Tester of the Month Awards !!!

❖ Winners for <u>Testing Crossword</u> :

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

❖ Winner for <u>Testing Puzzle</u> :

Name : Zeeshan Shaikh

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

Thirteen Months

10000+ Readers

89 Countries

ONE Magazine

TEA-TIME WITH TESTERS

Subscribe here Right Away to get our all Issues for FREE

Measuring Cost and Value (Part 4)

The Case for Specification Reviews

Background

The specification review for the Zebra project led to the listing of more than 150 serious issues. Management, when considering this list, realized that the product was ill-considered. They cancelled the project and initiated project Iris to replace it.

Development Costs Saved By Cancellation

Direct labor, 6 person years = $420,000

Project management = $82,000

Hardware resources = $60,000

Total development costs saved = $562,000

Marketing Costs Saved By Cancellation

Product manuals = $55,000

Training & development = $25,000

Sales literature = $30,000

Total marketing costs saved = $110,000

These figures assume that the product is brought to the point of product release before cancellation. Until this project, our company's historical pattern has never been to cancel a development project until at least the first scheduled delivery date. Later cancellation would, of course, incur additional costs, including field service training, attempted support, and loss of customer confidence and market image.

Cost of Delay in Coming to Market with an Alternate Product

Our marketing model says that a one-year delay reduces our market share from 28% to 21%. A two-year delay reduces it to 12%, and 3 years or longer shuts us out of the market permanently. The original market estimates for Zebra were $45,000,000 gross revenues, with a net profit margin of 30%. Thus, a 1% loss of market share costs us $135,000 in profits. A one-year delay would thus cost $945,000; a two-year delay, $2,160,000; and a three-year delay, $3,780,000.

Zebra was scheduled for a 1.5 year development period. Based on the model that an alternative product would be started when management began to get "signals" of trouble, the conservative estimate of the delay would be one year.

Based on our historical performance, the new product would slip its schedules for at least 6 months before management would cancel, yielding a two-year delay. Most typically, however, our projects that have been troubled by high change rates due to specification difficulties seem to run double their original development time estimates before cancellation, or in this case, three years.

Conclusion

The benefits from just this one specification review fall in the range $1,600,000-$4,500,000. Not all results will be this spectacular, but even the lowest of these amounts would pay the cost of about 2,000 reviews, even if they turned up nothing.

Sample Report for the Single Greatest Benefit Method

Issuing this report was a milestone in the cultural change of the organization. Previously, there had been no explicit cancellation process. Indeed, cancellation of a project was considered a rare exception, and a very bad event.

Discussion of the report led the managers to realize that more than one-third of their projects were eventually cancelled.

They developed a process that broke larger projects into five explicit smaller projects—expending approximately 1%, 2%, 5%, 20%, and 100% of estimated cost. At the end of each of those projects, management did a recalculation of cost and value, then used this information to make a decision on whether or not to proceed.

1.5.4 Example Report #4

Another example is based on a study Dani and I did some years ago. This study was initiated by a client manager who questioned the worth of experiential training for technical leaders. He claimed that training could not be evaluated, but was convinced by the study, which led to the inauguration of a technical leadership development program for a technical staff of 3,000 people.

The study used interviews to discover the greatest single benefit of the training to 100 people who had taken the training six months or earlier. We tabulated only one benefit from each person, using the largest one that they could remember clearly and document the value in a way that was convincing to the manager. In this way, we ensured that the estimated value was a lower limit, for many of the participants could document additional benefits. We obtained the following table:

Number $ Value of Total Return of people benefit on investment

1 person documented a benefit of $1,000,000 for a total benefit of $1,000,000

2 people each documented benefits of $500,000 for a total benefit of $1,000,000

7 people each documented benefits of $100,000 for a total benefit of $700,000

15 people each documented benefits of $50,000 for a total benefit of $750,000

20 people each documented benefits of $10,000 for a total benefit of $200,000

20 people each documented benefits of $5,000 for a total benefit of $100,000

10 people each documented benefits of $1,000 for a total benefit of $10,000

25 people could not recall and specific concrete benefit

Total return on investment = $3,760,000

Total cost = $200,000

Ratio of return to cost = 18.8

This study convinced the manager to inaugurate the training, but it convinced us that we no longer had to be apologetic or defensive when someone asked us to put a value on our work. Nowadays, we simply interpret the request as a reasonable desire for information on quality. This is the position a software engineering manager needs in order to be part of a Pattern 3 (Steering) culture.

1.6 Helpful Hints and Variations

1. Value cannot always be determined in monetary terms, because people hold certain values that are not directly convertible to money. The programmer's value system may include praise from peers for tricky coding, something for which the customer would pay nothing, or even pay to avoid. Managers may try to motivate programmers to adopt the customer's value system by mentioning how much money something is worth, but that kind of argument falls on deaf ears. Even offering bonuses for

certain tasks may not work if the programmer is simply not working in the monetary system of values. On the other hand, the programmer might do almost anything for the promise of a chance to work on a new system that offers stunning technical challenges.

2. With many customers, a small-value piece of software can have high total quality, even though the producers don't think it's a high quality item in their value system.

3. Quality is value. Nothing more. Nothing less. Quality is total value to all people over all time. Thus, if you lose users, you lose value. Time does not become a quality issue except in terms of the time/value trade-off.

4. Jim Batterson applies the Second Law of Thermodynamics to software maintenance, describing the tendency of software to decay of time without corrective effort. Lehman and others observed that successive changes tended to affect larger and larger numbers of software modules, reflecting a decay in design cohesion.

1.7 Summary

1. The motivation for improving quality always starts with a study of the value of quality, but many managers seem to confuse cost and value.

2. When under pressure to justify its existence, an organization has to decide whether to emphasize the cost side or the value side. The switch from cost observation to value observation is the strongest indication that an organization has made the transition from Pattern 2 to Pattern 3.

3. "Effort moves to what is counted." Cost counting leads to cost reduction. Value counting leads to value enhancement. Cost reduction is limited by the annual budget. Value enhancement is unlimited.

4. When someone says, "Software costs too much," it's always a coded message meaning, "Software isn't worth enough (to me)." Value is always perceived value, so we must know who is doing the perceiving.

5. When a software project simply collapses, the quality question is easy to answer—quality is zero. Underlying all quality collapses is the simple fact that in software we are attempting to achieve quality through precision higher than human beings have ever attempted before.

6. Software is a young, maturing business. What was an acceptable process for producing quality in the earlier system becomes unacceptable in its larger, more complex successors. Thus, even when software value doesn't collapse, it decays.

7. The Second Law of Thermodynamics says that you have to pay to get quality, and the higher quality you want, the more you have to pay. The First Law of Human Nature says that nobody wants to believe the Second Law of Thermodynamics applies to them.

8. The investment in quality must be more than money and hard work. To produce quality consistently, managers must learn new ways of thinking.

9. Systems thinking tells us that to produce quality, we must monitor requirements as they change, or as our understanding of them changes. Then we must make adjustments in the process on the basis of deviations between what is required and what is produced. That's the manager's job.

10. Measurement of quality through secondary requirements is indirect measurement, because it requires an extra step to relate the measurements to quality, and that extra step could go wrong. The possibility of misinterpretation means that the more indirect the observation, the more carefully it must be managed.

11. Discontent over standards arises when people who must conform to the standards cannot make the cause-effect connection between the standard and the value of the standard. The detailed impact case method and the single greatest benefit method allow us to make this connection and thus measure quality.

12. The detailed impact case method is an exhaustive study of the value impacts of a change. It is based on the idea of tracing requirements through a diagram of effects.

13. The single greatest benefit method is one approach to putting a floor estimate on value, while keeping the cost and time of the estimating method down to practical levels. The basic question of the greatest single benefit method is "What is the one greatest benefit that you can attribute to this change?"

1.8 Practice

1. Next time you hear, "Software costs too much," try transforming it into an investigation. Ask, "If software would cost this much less, how much more could we sell?" Graph the cost and the value, then choose the trade-off point that gives maximum net value. This is how much software ought to cost.

2. Choose some standard that has been the nub of discontent in your organization. Conduct a detailed impact case study for that standard, to see if you can determine why people think it doesn't provide value for the trouble it requires.

3. Many organizations are in the habit of announcing delivery dates before requirements are set. What message does this send? What is the effect on motivation?

4. Instead of announcing delivery dates for your next product, try announcing the quality level, as in the slogan,

"We will sell no wine before its time." What would be the effect on wine purchases if you announced, "We will sell no wine until marketing says we need some cash flow." Or, how about, "We will ship no software until the date the president set to satisfy the venture capitalists." Sometimes it may be necessary, but what will be the impact?

5. Dawn Guido and Mike Dedolph suggest that there seems to be an inverse correlation between the utility of a standard and the probability of successfully obtaining a waiver. What causes can you imagine for this effect?

6. Jim Batterson says, "The way I think that quality is free is that one can develop quality habits and habits of rigorous thinking so that one gets to doing things right the first time, and productivity is greatly increased. Where's the cost?" Brainstorm some of the costs of developing "quality habits and habits of rigorous thinking" in an organization that has already developed habits (i.e., a culture) that produced "quality" in another environment.

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.

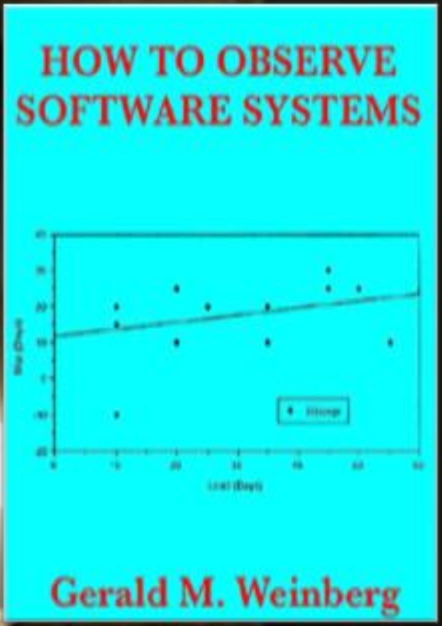
For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including The Psychology of Computer Programming. His books cover all phases of the software life-cycle. They include Exploring Requirements, Rethinking Systems Analysis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

HOW TO OBSERVE SOFTWARE SYSTEMS is one of the most famous books written by Jerry.



This book will probably make you think twice about some decisions you currently make by reflex. That alone makes it worth reading. "Great to understand the real meaning of non linearity of human based processes and great to highlight how some easy macro indicator can give info about your s/w development process." An incredibly useful book. Its sample can be read online here.

TTWT Rating: ★★★★★

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

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

Understanding the defect mindset

By Mike Talks

Defects. To many other roles on a project, they are a beast to be feared. But testers have a unique relationship with them, because through experience testers have learned to understand them and whenever possible to tame them.

On any project testers can expect to field many questions on defects.

Here are lists of some of the more common questions I've found myself dealing with from project managers and business owners alive over my career ...

As a senior tester, I sometimes find myself feeling a little bit like Bill Murray in Groundhog Day when it comes to fielding questions about defects. So much so that I ended up including some FAQ about defects in our department handbook.

No … I didn’t do this to rudely defer any questions from team members to the handbook, but instead to just polish up my reasons why, so I could give better, slicker and hopefully more informed responses …

What is a defect?

A defect is anything encountered in the system under test which is not as expected. It can range from *"shouldn't the font be red here not green?"* to *"why do I get a blue screen when I do this?"*

Testers are the creators of most defects, but anyone in the team can find them (although they might need a testers help logging them as defect reports).

Within a piece of software, it's likely that a lot defects will be raised – to help organise them, they are usually categorised by severity (how big an issue they are) and priority (how urgently they're needed to be fixed).

Defects, testing, quality

Testers test software to prove it is suitable for purposes, works as expected, and is ready for release. Along the way as they test delivered software against its original design and requirement, they will find defects. Defect reports are the value that testers add to a project, because they help to diagnose problems and improve the quality of software to be released.

I've never worked on a project where we didn't find defects. Turn that around and I've never worked on a project where testers didn't help to increate the quality of the product.

Why do we need a defect tool?

Most projects should have a defect tool of some kind for the logging of defect reports found. At bare minimum a defect spreadsheet should be used to keep tabs on issues.

Defect tools have the advantage of recording all important information (date logged, by who etc), but most importantly they allow a defect to be actioned against a person or group.

This allows defects to be much more easily managed, and help with the number which can be managed.
To be truly effective, everyone who might need to be involved with defect creation, actioning, and resolution needs to have access to the tool, so they can be actioned by it, and return defects to people when they've done work on them. Within our organisation, this is relatively simple, but there are challenges when this comes to raising defects against components from external parties who might have limited access to our internal systems.

Defect Lifecycle

The defect lifecycle is like a miniature version of the software lifecycle, dealing with only one feature.

- **Raised** – The defect is recorded by a tester.
- **Assessed** – The defect is assessed by test manager/project manager, and actioned against someone for some piece of work. The problem might be against the back end software, the GUI design, or even with either the requirement or the test itself.

- **Actioned** – whoever is actioned needs to make a change to their work, and check it.
- **Fixed** – the corrective work has been completed by the developer, BA or tester, and is ready to be rechecked.
- **Retest** – the defect is assigned to a tester to retest. If happy this can then be closed. If the problem's still there, we go back to Action or Assess.
- **Closed** – the defect is closed. *Hooray*!

What makes a good defect report?

Simplicity and repeatability.

Most importantly it should be a set of instructions which allow someone *(your developer ideally)* to repeat the problem for themselves. Ideally you should try it out a couple of times and be happy you can repeat the defect yourself. If it's a high level defect, you should really bring around your *developers (if possible)* to talk them through it.
But here are some other things which are useful to record,

- Build version tested. Because sometimes the problems we find are that we're using an older build rather than the latest to test.
- Severity. How much does it impact business if this went live?
- Priority. How urgent is it this is fixed (usually linked with Severity).
- Date. So you know how long the defect has been open.
- Why it's a defect. Refer back to original design/requirements to say why you think this shouldn't behave like this. As much as possible it's good to source defects back to requirements.

Do we need to defect everything?

Defect reports are a way to keep track of issues you encounter. Generally it is worth recording them. However if a developer thinks they can have something fixed within a day, it's worth holding off and waiting to see if he can fix it.

But generally it's worth recording them, even if you close them almost immediately. Even defects which essentially read as "software looked wrong, but on closer analysis of the requirements, it's working as specified" are worth keeping note of. Sometimes customers and project management encounter the same observation, and it's useful for their confidence to see a copy of the investigation.

I can't repeat an issue. Should I still defect it?

This is a toughie. Generally a defect you can't repeat has little value to either managers or developers. Everything depends on how severe it was.
If you were using an accounts screen, and had a blue screen which required the machine to be restarted, that's a high level defect, and should be recorded, even if you can't get it to happen again. If after a few weeks of testing it's not reoccurred, it possibly can be dropped from high to low severity (or even closed) after conference with managers because it doesn't seem to be happening again.

There might well have been something in the background of your machine (new Microsoft update for instance) which caused this. And even a closed defect can be reopened if it rears its ugly head again.

We're in week three of testing. We found the same amount of defects this week as in week one. The software isn't getting any better!

A common worry from managers.

It could be week 1 was slow because a couple of severe level defects prevented whole areas from being tested.

More often though you do find week-on-week similar numbers of defects do seem to be raised. This is where closer inspection of the numbers is important – remember there are lies, damn lies and statistics.

A trend in many projects is that most of the big defects, by which we mean the high severity ones, are discovered in the first few weeks of testing. As project go on more defects are found, possibly more, but they're of less severity.

The trend you're looking for when looking at defects found week-on-week is how many high or severe defects are being discovered. This should significantly decrease as testing continues. If not, then you are not yet over the "pain point" of your project.

We still have defects open – we can't go live can we?

Contrary to popular opinion, you can choose to go live with a product that has open defects. We have to live in the real world, and that means a drive to deliver.

What we aim to do is to go live with as few defects as possible. That there are no high or severe level defects unresolved (as this would really impact on our business). That the defects which are open are understood and more importantly the impact on the business is understood, and any workarounds communicated.

Why do we get defects? Who's to blame?

As software gets more and more complex, it's an inevitable that mistakes will get made throughout the software development lifecycle.

These mistakes can be due to simple human error, or can be because we design on part of a solution without understanding it will have an effect on another part of the system.

Defects are often associated as something testers raise against developers. But every part of the software development lifecycle are prone to unexpected defects. It's just when a developer creates code that those defects become most visible and impossible to ignore.

In reality defects can be due to,

- BA not understanding the process they're designing requirements for
- Architect missing a dependency in the system
- Developer making a mistake in code
- And yes even a tester misinterpreting the requirements

So everyone can be the cause of a defect, it's not just a developer problem.

We have multiple defects raised for the same problem …

This always looks bad. Generally you should always search to see if a defect has already been raised before raising it. Sometimes this isn't as easy as it sounds, because a system is failing in two different places but due to the same root cause (only found on investigation). To avoid confusion (and two programmers fixing the same bug), you normally can close or link one defect to another.

How can I make my project not have defects?

This is actually a trick question. There are always going to be defects in your system. Because this is human nature.

No matter how much you review requirements and sift through lines of code, you'll always miss something.
However the way to reduce the impacts of those defects is to try and find them early as possible.

That means

- Review requirements and design as they're written to make sure it flows for the likely business cases. Ask "what happens if" now rather than later.

- Don't just accept code as done because it compiles. Try to unit test now rather than wait until later testing to find issues.

- Get early releases of software to test to ensure the software generally feels right and is going in the right direction.

- If you can, automate tests where possible so developers can check new builds against tests to make sure the build is stable.

Following these guidelines means you'll find a lot of defects before they become big issues. If there is a flaw in the design, and it's only when you're doing acceptance tests, this is going to mean major rework – add length and cost to the project.

Reviewing, walkthroughs, unit testing, prototype testing shouldn't feel like they're impeding progress on a project. In fact they're vital for weeding out problems early.

At the end of the day, commitment to quality shouldn't be a tester "thing". It should be a whole team "thing".

What is the standard definition of defect severity?

As discussed, defects will be categorised according to their severity.

This is the standard categorisation of defect severity using within our department. It is expected that testers will triage defects according to severity, and inform project managers within an hour of finding either 1 or 2 level severity defects.

Severity	Name	Responsibilities
1	Showstopper	*The defect would have catastrophic consequences for Our organisation if the system went live.* Defect *causes*; • System crash • Massive loss of performance • Corruption or loss of data • Security violation
2	High	*The defect would cause major embarrassment to Our organisation if the system went live.* Defect *causes*; • Operational error • Data integrity issues • Some performance degradation • Loss of functionality for which there is no work around. • Any issue that could cause our organisation major embarrassment if product went live with this problem. • System reboot • Textual error on item that is a legal obligation
3	Medium	Defect *causes* a loss of functionality, for which there is a workaround.
4	Low	*The defect represents a nuisance to Our organisation and its customers if the system went live.* *Defect causes;* • *System works as expected, but is confusing for user*
5	Text	There is a textual error in the system (spelling or grammar) which does not cause confusion and is not a legal obligation.

The defect classifications will be mitigated or escalated by common sense, for instance a system crash (1) which happens only under very rare conditions could be considered as a Low (4) because we're not really worried about it occurring during normal circumstances. Whereas a typographical error (5) which makes a swear word would be elevated to (2), as it would offend our customers and embarrass our organisation.

Mike Talks is the chief tester for Payment Services at Kiwibank in Wellington.

An ex-programmer – but don't let you hold that against him. He's worked for 16 years testing projects ranging from complex aircraft avionics to simple websites.

Cursed by gypsies at an early age, he finds himself invariably breaking anything he touches. Nowhere more so than when he tries to use technology.

He is on Twitter as TestSheepNZ

Back To Index

Coming soon ….

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

Coaching Testers

By Anne-Marie Charrett

This is a coaching session I had with David Slick on the topic of Risk Based Testing. It's a good example of the how I run a coaching session. An important part of a coaching session is the evaluation of the transcript **after** the coaching session take place to help re-enforce learning.

I like this example of a coaching transcript because its shows how evaluating a transcript can help the coach improve as well.

I've highlighted aspects of the coaching session and provided some comment to explain what I was thinking at the time, and what I was trying to do. I often refer to Patterns in my coaching. These are typical coach or student behaviors that are notable either because they happen a lot or they're significant to coaching.

Previous to this coaching session, I'd asked David to send me an email with an analysis of his strengths and weaknesses and also to describe any challenges. He made the following comment:

> "It's important to me to focus on value to the customer, but at times, I've definitely gotten off course. There are many bugs that our team has missed because our risk assessments have failed to identify something that we should have. I guess I'd say my risk assessments have been very informal and not thorough enough (due to failure to allocate time to the activity when getting caught up in tight project schedules)."

I've edited and removed parts of the transcript to keep the article concise.

Understanding the Student

Anne-Marie Charrett: The last time we spoke we talked about assumptions and we drifted into oracles we can touch on that today or we can go into risk in testing

Pattern: Hand over the reins

One of the essential patterns in my coaching is to put the tester in charge of their learning; the coaching becomes more applicable to their context.

David Slick: how about risk?

Anne-Marie Charrett: so, what's your definition of risk?

Pattern: Common Ground

My aim to get a common understanding on risk, though perhaps a more appropriate question would have been to ask, "What is risk based testing"?

David Slick: So maybe risk is something that threatens value to something, could be the customer or the project or my job. The standard thing I've seen (and used) to define risk in a quick and dirty nominal way is to do chance of something occurring (1-3) * impact it would have if occurring (1-3)

Anne-Marie Charrett: I call that risk based test management - where you *manage* your testing according to the risk it has. Risk being defined by the impact of failure & the probability of failure[1]

What I want to look at today is risk based testing

Most testers can identify Risk Based Test Management but not all understand Risk Based Testing. I wanted to clarify the difference.

The Task

There are two tasks here, the first is observation and the second is risk based testing.

Anne-Marie Charrett: I'd like you to go to this website http://members.iinet.net.au/~pontipak/redsquare.html and tell me what you see.

Diagnostic Pattern: Observe the Observer

You can only test what you 'see'. What does David see?

David Slick: I see a black square outline with an open white space containing 4 smaller blue rectangles and a red square in the middle of them and some instructions below. Two links and a bunch of white space and the browser of course

We discuss different ways to model our software.

Anne-Marie Charrett: What I'd like you to do is to make a list of possible failures

One way to approach risk based testing is to create a possible list of failures and test for those. Testers in general find it hard to define and abstract out possible failures until they see them. It's a good practice to work out the failures before testing; it generates a different model for testing, which then helps to create new test ideas.

It might have been a good idea for David to explore what a failure is. What the hell, sometimes you just have to jump in!

David Slick:

- There might be something wrong with the timer
- Might be able to escape the boundary of the black square and "cheat"
- The click-n-drag might fail
- The squares may not move in a predictable manner, which would be fine if that's what you wanted it to do, but it seems the game would be a lot less fun if that was the case
- Clicking outside of the boundaries of the square may result in erroneous errors (which happened)
- Scalability

Anne-Marie Charrett: Let's take a look at one of your failures - scalability: tell me what the failure is

Pattern: Driving to Detail

I use this pattern a lot, when I ask testers to be more specific about a concept. Here I'm asking him to be more specific about the failure he is describing. What exactly is the failure he's trying to generate?

[1] G Weinberg in Perfect Software

David Slick: the failures related to this topic would be failure to present the page after a certain number of users hit it relatively close to one another OR poor performance when multiple users are hitting the page simultaneously

Pattern: Drive to Detail, *this is not specific enough, I try again.*

Anne-Marie Charrett: Let's take the first one **failure to present the page** after a certain number of users hit it relatively close to one

I should have used the ***Terrier Pattern*** *here, and not let up on the topic of failure. He hasn't abstracted the failure out enough to generate different test ideas, they're still linked the concept of the number of users.*

Anne-Marie Charrett: and write a couple of test ideas for it

David Slick:

Quick and dirty - hit it simultaneously with multiple browsers, hitting F5 with both of my hands on two different machines over and over again for a couple minutes.

advanced - code up a multi-threaded application (could use some performance tool like loadrunner or silkperformer) that simultaneously hits the site, crank it up until it breaks or you stop caring, which doing the first test to see what the user experience is while this happens and detecting in the coded check whether or not you get a response from the server

Anne-Marie Charrett: They are the same test idea though

Here I'm challenging him on these test ideas. They're very similar. Normally I'd challenge him to come up with some different ideas, but we ran out of time. Instead I supply him with some ideas I had.

Here is couple of ideas I had:

1) Use different CPU's to slow the machine process down

2) Throttle the bandwidth

3) Slow the Mouse down (I'm not sure if this would do anything)

Debrief

This debrief here is a little short. Normally we review the whole session, highlighting insights and re-enforcing learning. I need to learn to manage my time better in coaching sessions!

The more precise your failure-list the better your test ideas will become.

Your failures seemed a little vague.

Go through your list and ask yourself what exactly is the failure here? Would I be able to recognize it?

The session continued in email format, with David coming up with a great list of test ideas by concentrating on the failure, not the cause of the failure.

The key points to remember when coaching testers are:

- Respect the tester. Make the effort to understand them and keep the session relevant to them. The exception here is a diagnostic coaching session where you're evaluating a tester's skill level and mindset.

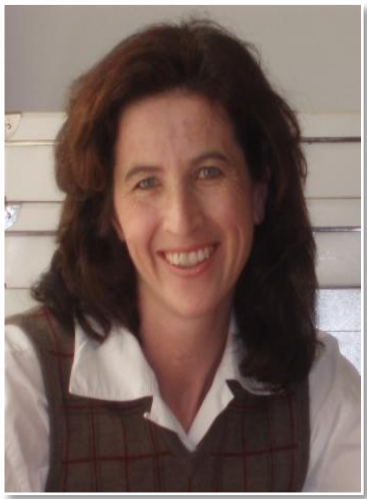
- Practice transpection because as the coaching is taking place, you need to be having a 'second conversation' with yourself about the tester by asking yourself questions such as: "What's missing", "what may they be failing to understand", "how would I have answered that question"?

- Avoid the temptation to dish out advice. This is not a consulting session. It's a coaching session. Use Socratic dialogue to help you in this. Question the tester, and don't be afraid to go into detail.

- Take your time, slow down the coaching process. That's the great benefit of coaching on Skype, it forces both the coach and the tester to slow their thinking down. This helps understand the thought process behind the decisions being made.

Many testers leave coaching sessions feeling invigorated and enthusiastic about testing. That's the way it should be. Keep your coaching sessions positive by highlighting insights and learning.

Back To Index



Anne-Marie Charrett is a testing coach and trainer with a passion for helping testers discover their testing strengths and become the testers they aspire to be.

An electronic engineer by trade, testing discovered Anne-Marie when she started conformance testing to ETSI standards. She was hooked and has been involved in software testing ever since. She runs her own company, Testing Times offering coaching, **workshops** and consulting with an emphasis on Context Driven Testing.

Anne-Marie describes herself as Iristralian having been born in Dublin and now residing in Sydney, Australia. She happily engages in work in both the Northern and Southern Hemisphere.

Anne-Marie can be found on twitter at charrett & also blogs at **http://mavericktester.com.**

~ 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

3. Facilitating a Journal Club

The focus of this article is creating a journal club which defines strategies that a manager can implement for his department. Testers can also create a journal club to further their own professional development.

A tester needs to be mindful that skills could become stagnant and there are many ways to keep those skills sharp. One way this can be done is through reading articles and blogs. This should be an important part of a tester's life to ensure he is staying with current trends. Understanding how other professionals are approaching testing problems can challenge and progress skills and strategy. The following sections provide suggestions to facilitate a journal club.

Facilitators

A journal club can be facilitated by the Software Testing Manager who will oversee the distribution of articles, review schedule, and discussion. The manager may consider allowing his team to have a larger role by assisting in the review sessions and identifying articles. If the testers create their own journal club, then either one tester can be in charge or they can rotate the responsibility. (Going forward the term "facilitator" will be used to represent the person who is in charge of the journal club.)

Meeting Rules

Typically there should not be many rules since a journal club should be an open discussion about an article. An important expectation to set is that everyone reads the article and is prepared to discuss. If necessary, ground rules can be defined based upon the personality types and number of testers. Some journal clubs only allow articles written within a certain number of years. Be careful with this type of rule because older articles can provide timeless and valuable information.

Selecting Articles

The facilitator may identify the articles; however it is a good idea to ask the testing team to contribute articles in their area of expertise such as mobile testing, load testing, and exploratory testing. Distribute the articles 1 – 2 weeks before the journal club meeting. Below are guidelines to help in the selection process.

- Identify a variety of articles / blogs of actual testing practices that can be adopted by your team.
- Identify articles that are theory-based but challenges the testers to think differently.
- To mix things up a bit, introduce webinars to watch before the meeting and then discuss the material.
- Select a few similar articles that may support or contrast each other.
- For teams that want something more challenging, introduce book reviews or have each tester read a chapter to present back to the team.

Starting and Progressing Conversations

The facilitator starts the journal club meeting and often needs to "break the ice" to encourage the testers to discuss the article. Below are a few suggestions to start the meeting.

- Provide a brief overview of the article.
- Provide a starting point with a particular item in the article. Ask for comments or go around the table and ask for feedback.
- Ask what and how you would apply something from the article.
- If the conversation starts to fade, ask what else they found interesting or would like to discuss.

Initial Journal Clubs

Sometimes people are reluctant to participate during the initial journal clubs. This can be contributed to the fear of sharing an opinion that might be different than the other testers; not being confident in what to contribute; and just not sure what this is all about.

For the first few sessions, monitor who provides opinions and takes a leadership role as they may be able to help you progress the journal club. To foster more discussion, ask a few open-end questions and go around the table for input or direct the questions towards the nonparticipating testers. For example:

- What do you agree with in this article?
- What do you disagree with in this article?
- How would this approach help you in testing?

When closing the meeting, reiterate the purpose and provide guidance to encourage participation in future sessions. For the testers who do not participate, try to discuss an upcoming article with them individually to help them determine how they can participate. Sometimes they will tell you that the other people already viewed the same opinion reducing their opportunity to participate. In those situations, call upon him at the start of the journal club to express his opinion. Try different approaches to help testers find their voice and confidence to participate; however maintain the delicate balance of participation versus intimidation.

Managing Strong Opinions

Lively discussions are great; however, reign in discussions that are not helpful to the team. If one person is dominating the meeting, you need to take control of the meeting by either changing the topic or including more people in the discussion. This can be a difficult situation because you want them to freely discuss their opinions; however the journal club must be beneficial to the testing team and the time allocated for the meeting.

Some approaches to manage the meeting can include:

- Thank the tester and then guide the discussion to a more beneficial topic.
- Thank you John. Now let's discuss a different topic from the article. Mary, what were your thoughts on this topic? (With this approach guide the discussion to a different topic and select a tester who is not shy or timid to continue the discussion.)
- Expand the discussion to the group by asking for their opinions.
- Now that we have heard from John, what other opinions are there on this topic?
- Report how much time is left for discussion.
- We have 10-minutes left of our meeting, lets use the remaining time to go around the table for any final comments. (With this approach start with the person next to John to allow each person an opportunity to comment.)

Frequency of the Journal Club

How often to hold a journal club meeting depends upon several factors. One of the more important factors is to ensure it does not become a burden to the team where the time spent versus value-gained is not recognized. Unless the team wants a shorter-schedule, a once a month or every other month meeting should be sufficient. The length of the discussion can go from 30-minutes or longer depending upon the number of testers and their contribution level.

Try to maintain a schedule to ensure the journal clubs do happen. However if the team is working towards a tight timeline or there is not a good article to review, then skip a meeting. Just be careful not to make this a habit or the journal club will phase out.

Tips

- Start small and build upon the number or complexity of the articles.
- The journal club should be a safe environment. For testers who are reluctant to participate, work with them individually to help them find a way to contribute.
- Rotate the responsibility of facilitator and encourage the testing team to submit articles for review.
- Provide a variety of articles, webinars, and book reviews to keep it fresh.
- A journal club should be a positive experience. Find a happy balance between frequency and number/length of articles to read.
- Try to maintain a regular schedule; however when there are pressing deadlines, consider canceling the journal club meeting to reduce stress for the team and review the articles at the next scheduled time.

Conclusion

When managed properly, a journal club can be a great way to introduce different testing approaches to solve problems and expand knowledge. A journal club can be facilitated by the testing manager or that role can be rotated among the testers. Reviewing a broad range of topics relating to actual testing practices and theory is beneficial to keep the discussions interesting and fresh. Encourage participation by working with testers who are reluctant to share an opinion to understand their reluctance and how they can find their voice. Overall the journal club should be a positive environment where opinions are exchanged, testing skills are challenged, and new techniques or ways to optimize testing are implemented by the testers.

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

THE OTHER SIDE OF BUCCANEER SCHOLAR

James Marcus Bach.

A guy who left his schooling at early age and started as video game programmer is now an esteemed software tester, author, trainer and consultant.

He is a proponent of Exploratory testing and the Context-Driven School of software testing, and is credited with developing Session-based testing.

How did he achieve this?

What made him to break the laws?

What does he think about Software Testing?

What message does he have for young testers?

Find it out in his exclusive interview with **Tea-time with Testers.**

Read on....

Before we start, we would like to let our readers know, 'How did things start off for you?". "How did you land up in Software Testing?"

I started as a video game programmer after I quit high school. I found I didn't like programming that much. After a little while, I began to be bored and restless. There wasn't enough stimulation. I need social interaction. I need lots of variety in the problems I get to solve. It got so bad I became unable to work: I would get physically sick if I sat down to write code.

This was a disaster, because the only thing I knew how to do was write software. Fortunately, I heard about a job as a test manager at Apple Computer. I was amazed that people could get paid to test things. I got that job and never looked back. I still write software. I create automation to support testing, occasionally. I like programming for short periods.

Can you please help our readers understand 'focus-defocus' technique & how can it help them to test better?

This is a very broad technique that is used as part of the Rapid Testing methodology. All testers, all the time, are testing against and within some kind of model. We call that focusing. Maybe you are doing requirements-based testing. If so, then you are focusing on requirements. You have a specific idea what those requirements mean. Focusing is good. It helps you make sense of things. But to find a lot of bugs, you need to de-focus on a regular basis. That means changing your model, considering very different meanings of requirements, randomizing your tests, etc. To de-focus is to change your focus in a big way. Basically, all of testing can be seen as various kinds of focusing and de-focusing.

When you are confused about the behavior of a product, you should focus more. When you need to find more bugs, and you feel frustrated that you aren't finding enough, you should de-focus.

Why do you consider testing as a craft work? How differently do you see it compared other fields?

The word "craft" emphasizes the design aspect of testing and connotes a dash of discipline, too. I also call it an art, from time to time. Testing as a field is a lot like how surgery was 300 years ago, before smart people began flocking to it and turned it into something respectable.

We still have people peddling "best practices" in testing that are similar to the Galenic medicine quackery of the 18th century: unhelpful or else actively harmful folk practices promoted by authorities who lack scientific skills and ethics and discipline.

What is your view on major changes in software testing field likely to come in next 5-10 years?

The only change I can speak to with authority is that the community of erudite, skilled Jedi testers is going to grow.

These are people who vigorously practice their art and challenge each other to get better. These are the people who eagerly read social science textbooks and compete (in a friendly way) to apply the lessons to testing.

I hope that we will displace the old wheezy consultants promoting those silly practices I mentioned above.

I believe we already have begun to do that. It's a slow process, but I think ISEB and ISTQB will be quietly consigned to the history of stupid ideas.

The people who promoted them will probably declare victory, but eventually have to evolve or be made irrelevant.

Most of the managers ask for result in numbers e.g. more number of test cases, more defects etc. What's your opinion on this number game? Does that really matter?

It matters a lot. So, don't do it. It's unethical to fake your work. Counting test cases is simply a way to mislead people about the status of your testing. I stopped doing that in 1992. My refusal to count test cases is now old enough to vote in the United States.

Management doesn't count "management cases" to evaluate other managers. We shouldn't count test cases.

As a consultant, how do you pursue the quality of product? Is it similar to the way product/quality manager sees it? E.g. some managers prefer on-time delivery over quality of product. At times in certain cases quality doesn't matter to them but the delivery. Does that upset you?

I'm not concerned about quality, really. I don't control the ship date, nor do I want that responsibility. As a tester, my concern is about knowing the status of the product. I care about quality because my clients care. But if they don't care, part of my discipline is to also not care. However, I often have information that I think will *cause* them to care, and I eagerly share that with them. (Deep down, I want to be proud of the product, and I want them to feel that way, too, but officially, a tester is there to inform his clients about *their* concerns, not to promote his own vision of quality).

My clients want to know what kind of trouble the product may cause, or may have within it. I help them discover that.

I over-heard two guys discussing about pending tasks in their project work. One of them said , "Ohh Testing??? That can be done by any one. Don't worry about it." A lot of people think this way. What's your view? Can testing be done by anyone or it needs some skills to master?

My career is built on the idea that skilled testers, like skilled soldiers, do their job better than a disorganized and untrained rabble. But just as anyone can swing a sword, anyone can find a bug. Skilled testers will find more bugs, better bugs, and they will be able to explain and defend their work.

Anyone can find a bug. Only skilled testers can consistently find the great bugs.

How tangible do you find testing standards in real world scenarios?

Testing standards are never tangible. They are ideas. Paper documents are tangible, but it's the ideas that matter, not the paper and ink.

Perhaps you meant to ask how practical are testing standards. Well, if you are referring to the IEEE testing standards, I consider them harmful. I recommend that you ignore them. I was on the working group of IEEE 829-2008, and I repudiate it. Cem Kaner was on the same committee and he had his name removed. The revised standard was not developed in a collegial fashion, nor do I feel that the people who controlled that process were competent to talk about what anyone should do to document testing.

The upcoming ISO testing standard suffers from a similar problem. The political operatives who run that show don't care to face or resolve the actual problems and controversies. I have heard only terrible things about it, and I'm quite confident that the final standard will be a monument to the stupidity that is still tolerated among the gray heads of our field.

Do you believe that processes help average people to do things better? E.g. formal education for average kids. Is genius an exception? What made you to leave your schooling?

Process is a word that people use in mystical ways. Let's be more specific. The word process means "how things happen." Are some processes better than others for helping people develop their minds in the way they wish to be developed? I am confident that is true. But no, I don't think that institutional education as currently practiced by public schools is good for average kids.

I think it works for a very small minority of kids. For most kids it's a sick waste of their time and energy. In fact, it's a system that is obsessed by stratification and competition. It is actually designed to assure that very few children "get ahead" by defining success in narrow terms and heaping scorn on anyone who doesn't fit that model.

I would prefer that *all* the money used for "educating" children were used instead for local day care centers and open-to-anyone educational resource centers (we can call them "public libraries")—both of which would be optional to visit, rather than mandatory.

I agree that public funds should be used to help educate the public, but I also believe that in a free society education is a personal matter that is always be under the control of the learner himself, not matter what age he might be. The name for this policy, if you are interested reading more, is "unschooling."

I left school because I was wasting my time there. My father urged me to get on with my life, and so I did.

I thank my father for inviting me to become an adult.

James with his father Richard Bach

What is your take on becoming a certified tester (any certification)? Is it a must have thing to prove one's testing abilities?

Commercial certifications are blight on our craft. Please avoid them. Please join me in ridiculing and denouncing them.

When our craft decides what testing is, then we can talk about having a craft-wide certification. At this point in history, the most passionate and recognized people in the testing field do NOT agree.

Rex Black just wrote a little piece about how lack of metrics was the biggest weakness he sees in testing organizations. I think it shows he doesn't understand what metrics are or how dangerous they can be.

That's a pretty big gap between two people who are both so well known as "experts."

So, forget certification.

If you are ambitious as a tester, I suggest that you develop a positive reputation. For my kind of ambition, I need a public reputation. I run my business and charge what I charge based on the fact that I am well known all over the world of testing. I am not well-respected everywhere, due to the controversy surrounding the various schools of testing thought. But that's okay because there are enough admirers of the Context-Driven way of thinking that I have plenty of business.

My question to any young tester is what are you doing to study your craft, demonstrate your growing expertise, and publish your ideas?

If I am hiring a tester, and I Google a candidate for that job, and I find nothing that he's written online, he is at a great disadvantage compared to a testing blogger.

When you are not testing, what can one find you doing?

Reading! There are so many philosophy and science books that I need to get through. I'm so glad I have a Kindle. When I'm home I also watch a lot of videos with the family and pet my lovely dogs.

Unfortunately, I'm not at home very much.

Do you teach your son? How different is James Bach in testing conference and James Bach at home?

I am quite different at home. This is because my wife forbids me from thinking like a tester when I'm at home.

Well, let me put it this way: I can do what I like, because I'm aggressive and I'm a full-grown alpha male in my own house… but when I speak a certain way around my wife, she's sad. When she's sad, everybody in the house is also sad. In any marriage, a man can't be happy if his wife is not happy. So, I speak softly at home and I smile a lot and I drop my professional skepticism around Lenore. She has trained me well after twenty years.

When I'm at a conference, I'm on duty. I'm sharp and I don't easily accept what I'm told. This is how I think all testers should be at conferences.

I don't teach my son, Oliver. I behave normally around him, and he observes me. This is the way children learn. I find that overtly trying to teach my son makes him hate to learn. So he learns from me, but I don't teach him. Nobody does. Oliver has not been to school since he was 12 (he just turned 18).

Oliver and I recently visited BioWare, the people who create the Mass Effect video game. Oliver is extremely knowledgeable about video games, and it was great to see him test himself against the fine minds at BioWare.

(Incidentally, BioWare has one of the smartest and most motivated teams of testers I have come across. They reminded me of the team I taught at the Jet Propulsion Laboratory.)

James' family: Lenore, Oliver & Shasta

They say that 'school drop-outs' often develop a legendary path. What do you feel about your journey so far? Satisfied or still miles to go...?

To develop a legendary path, you absolutely must depart the ordinary path. From a young age, I was fascinated by mysteries and quiet places and forbidden choices.

I was repelled by conventional thinking.

It's a good thing I am not also a complete fool, or else I would have gotten into drugs and been arrested. Fortunately for me, my distrust of authority didn't manifest as violence or petty crime.

It came out instead on the intellectual plane: I don't trust teachers or experts on anything important unless they first earn my respect. This is why I had to leave high school. This is why I had to reinvent testing.

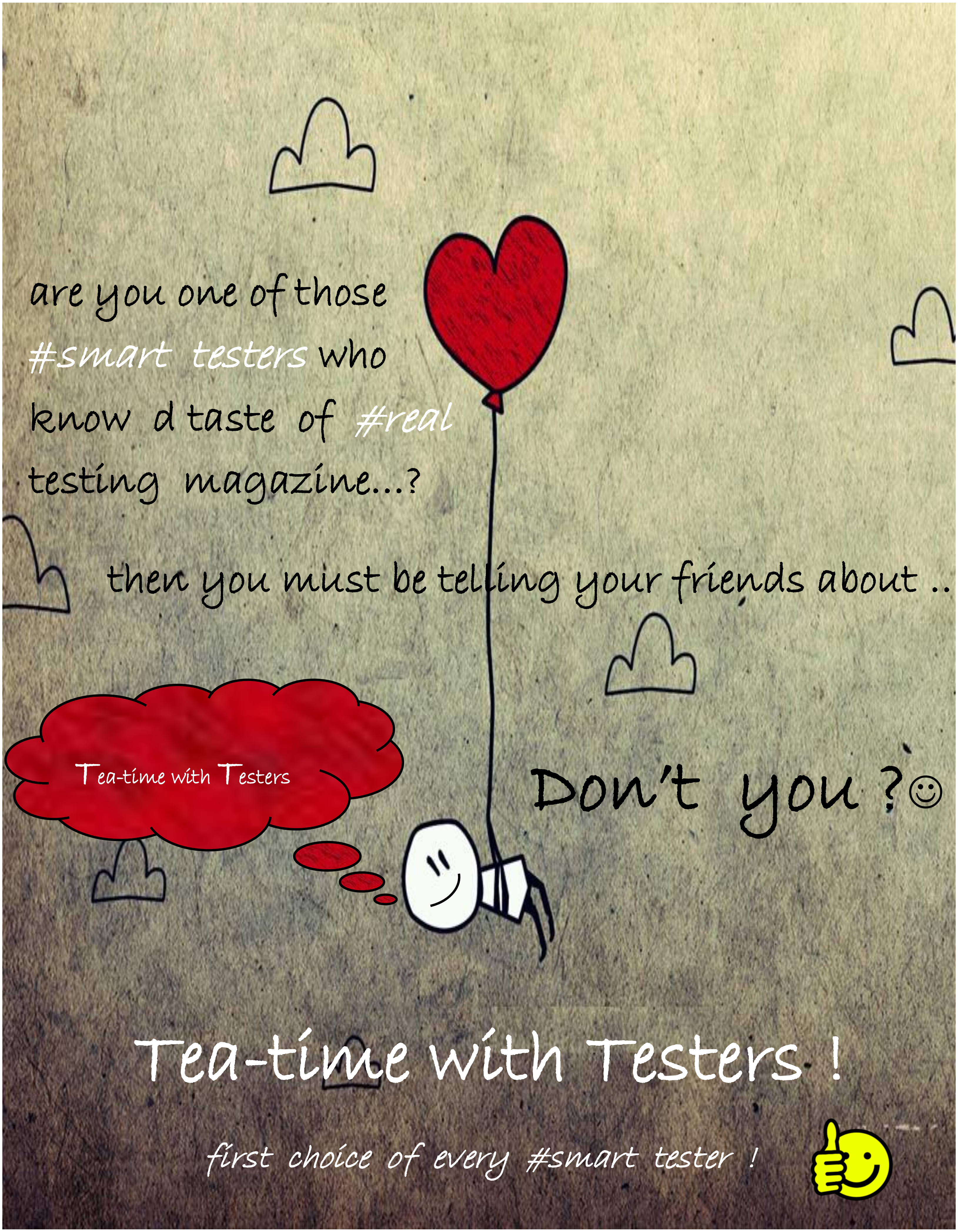
This is why it takes a teacher like **Jerry Weinberg** to hold my attention, and also why Jerry is a role model for me as I develop myself into the best testing teacher I can be.

Your opinion about *Tea-time with Testers*? We would be glad to have your feedback.

I'm generally impressed with the caliber of the people that you get to write for you.

That's why I'm writing for you, now. I want to be with the cool kids.

Back To Index


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**

Don't waste time on a Team/Company Quality Agenda!

Do you like wasting time on **useless tasks**?

Is your schedule so "Free & Open" that you are considering embarking on one or two **hopeless adventures** that will not add any value to you, your team or your company?

I am guessing (*even hoping*) your answer to both these questions is NO.

So think hard before you answer this next question:

Should you define and promote a Quality Agenda for your development and testing teams?

There are 2 quick answers, the "automatic" and the "pessimistic".

The automatic answer –

"DEFINITELY YES!!!"

I am in charge of Quality, and as part of this role I need to have an agenda and push it with all my strength.

I will fight for it and improve the way my company works! No matter how much people want to cut corners or make risky decisions I will make sure we work **only** by the rules!"

Then we have the **pessimistic** one –

"WHY BOTHER???"

In the end, no matter what we all say right now, the product will be released when Marketing wants to release it.No one will care if we have 3 or 7 or 17 showstoppers open in the system. Regardless of what I think or say about Quality we will keep working in the same way we do today.

So I better concentrate my efforts and those of my team on testing the product. Let's stop wasting time on useless battles…"

What is a Quality Agenda?

Simply put a Quality Agenda is composed of 2 main parts:

1.How your company (or your team) defines Quality in your product or service.
and
2. The actions or approaches you will take to increase the quality level of this work.

In the "Corporate World" this agenda is sometimes stated as a 2 or 3 line paragraph that hangs on a hallways of your building, or is shown as part of a presentation given by the CEO at the beginning of the year. Some times it is also handed out as small plastic cards that employees quickly place in their drawers (together with all sorts of company vision and mission statements from previous years).

If you will be writing an agenda that will be stuffed in a drawer or forgotten the minute after your CEO moves to the next slide, then I suggest you think of better ways to spend your time.

An effective Quality Agenda should be S.M.A.R.T.

(I think I wrote about this acronym in the past, but I find it so useful that I will repeat it's meaning once again.)

A good Quality Agenda, one that people will find useful and that will have a (positive) impact should be **S.M.A.R.T**. :

- **SIMPLE**- Something that explains in clear words what you want to achieve.
- **MEASURABLE** – You can clearly see if you are on the right track and making progress or going backwards.
- **ACTIONABLE** – Everyone should understand what to do and what not to do to advance the quality goals of the company.
- **REPEATABLE**- When talking about the progress metrics, they need to be clear and repeatable. Everyone who looks at them will be able to understand them and reach the same conclusions.
- **TIMELY**- The actions, objectives and goals should be based on the current status of the company, and need to be revised or changed based on the changing reality of the company and the competitive environment.

Is this really your task?

Now you understand what is a (good) Quality Agenda. This means that is time to start thinking whether this is something you should be leading within your company, or maybe stop from taking part on battles that are not yours to fight…

In my mind, when we talk about a Quality Agenda and we define it as the goals, plans and tasks aimed at improving the quality of your product and process, then **this actually is your task!**.

It is true that you are not responsible for the direct quality of your entire product, after all this task needs to be shared by all the team. But for me, the Quality Agenda is the responsibility of the QA Manager, since he/she should lead the process to achieve a better way to work and deliver higher quality products or services.

Defining the Agenda is only the start, the secret is to make this an on-going task

Defining the agenda is only the first part of your job. Once this is done you need to provide visibility and guidance to achieve the goals you set.

I said that good goals should be Measurable and Actionable, so it also your responsibility to make sure people are performing the right actions and that you are measuring the desired change in the system.

The best way to do this is to set up a recurrent event, it can be a meeting or even a monthly email, where you will give visibility into the progress achieved (or not!). I personally like the approach of setting up a *kitchen monitor* with this information.

You want to achieve a healthy competition between your teams that will push your quality forward.

In the past I've worked with monthly trophies to the team who made the biggest improvement and even with prices such as dinners and gadgets to the individuals who had the biggest effect on improving the quality of the whole company or process.

When should you NOT WASTE YOUR TIME on a Quality Agenda?

For me, this is a simple question.

If I feel that my management doesn't stand behind me and that they will choose to compromise quality in order to fulfill other business goals, or if I see that the goals we want to set as a company are not serious enough or not achievable by any means, then I will choose to make better use of my time.

Quality Agenda != Quality

Remember that you can have quality without having a Quality Agenda. The agenda should only help you to focus all the company employes, making sure we are all on the same page.

If your company chooses not to invest in the quality of your product and process, no Agenda or any other gimmick will help you to push this forward.

What's your take on this?

Do you have any success (or horror) stories related to the Quality Agenda in your company? Share them with the rest of us !

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

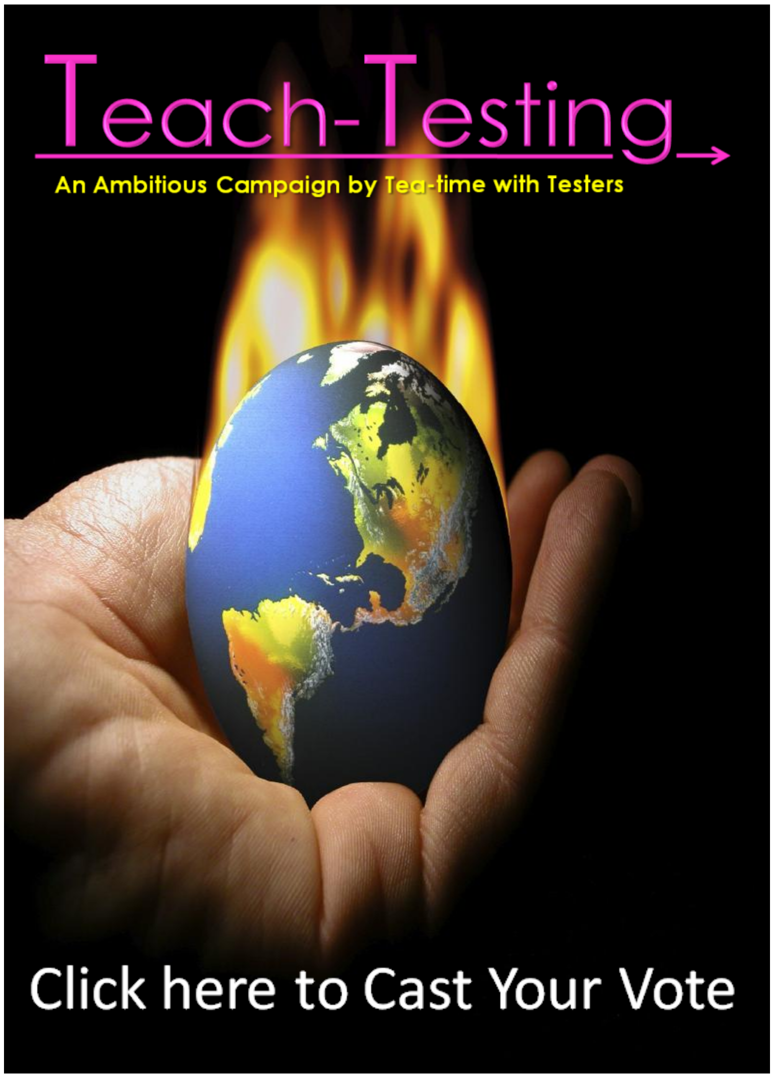
He's worked in companies ranging from small Internet Start-Ups and all the way to large multinational corporations, including Mercury Interactive (currently HP Software) where he managed the QA for TestDirector/Quality Center, QTP, WinRunner, and additional products in the Testing Area.

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

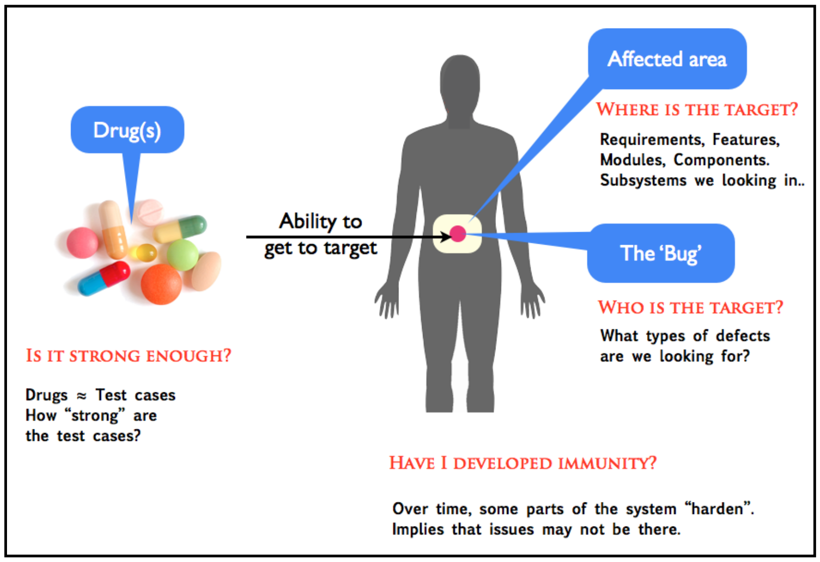
Do you know the "potency" of your test cases?

We all know that good test cases are the key to effective testing. So what is "good"? And how do we know if test cases are indeed "good"? When I ask these questions to test professionals, they say that a deep knowledge of application domain is required to answer this question.

Let us examine a non-software problem and seek inspiration to answer this question in a scientific manner.

How do we know if a drug that we take to cure a disease is indeed good? If the drug targets the specific bug , targets the specific area without affecting others and gets to that area as quickly as possible to cure the disease, then we think that the drug is effective. We say that the drug is potent.

Do understand that the drug potency can be markedly reduced if I develop a resistance to it.

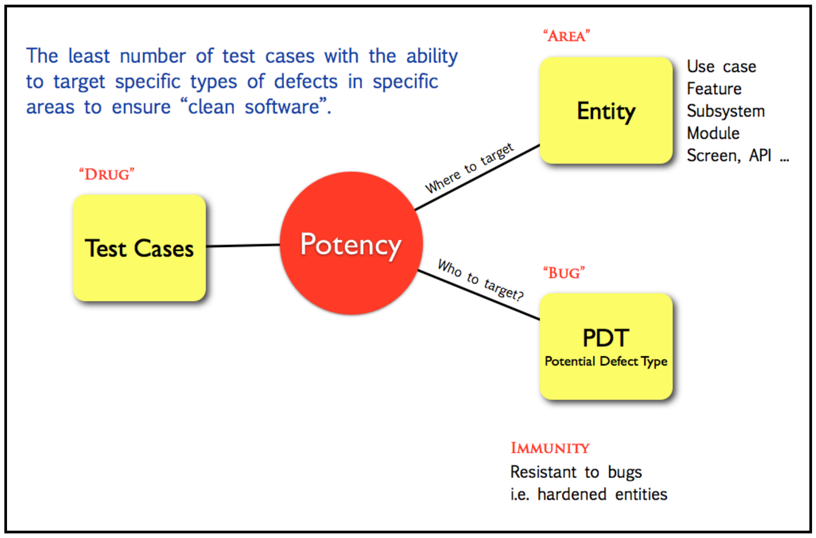

Drug(s)
Affected area
Where is the target?
Requirements, Features, Modules, Components. Subsystems we looking in..
Ability to get to target
The 'Bug'
Who is the target?
What types of defects are we looking for?
Is it strong enough?
Drugs ≈ Test cases
How "strong" are the test cases?
Have I developed immunity?
Over time, some parts of the system "harden". Implies that issues may not be there.

Note the similarity - test cases equated to the drug, potential types of defects to the bug and requirement/feature/code module to the affected area. So how can we use this example to assess the "potency of test cases"? A definition of "potency" is "the strength of the drug, as measured by the amount needed to produce a certain response". Potency is associated with efficacy of an entity, the ability or capacity to achieve or bring about a particular result.

In the context of testing, potency could be defined as - "the least number of test cases with the ability to target the specific types of defects in specific areas to ensure clean software". We could assess potency by examining four aspects:

- Strength of drug (aka test case)
- Area targeted
- Target bug i.e. potential types of defects and
- Immunity.

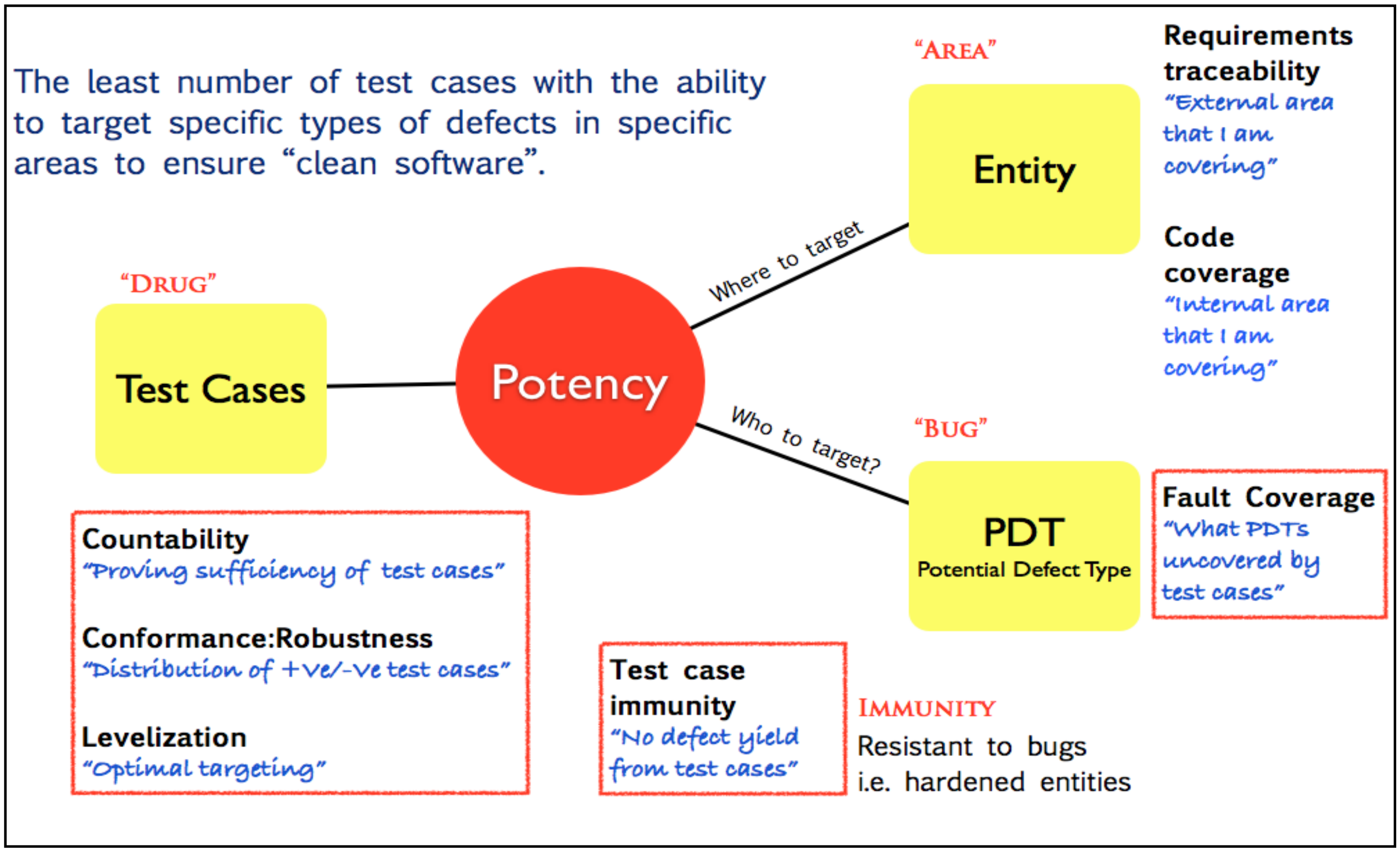
Examining properties of these aspects in a scientific manner could help us assess potency.



Let us dive in...

If we know that the test cases are adequate, that it covers the area under focus and targets those types of defects that matter then it is a great start.

Then examine the notion of "immunity" i.e. Could the system under test be resistant to the test cases?".


The least number of test cases with the ability to target specific types of defects in specific areas to ensure "clean software".
"AREA"
Entity
Requirements traceability
"External area that I am covering"
Code coverage
"Internal area that I am covering"
"DRUG"
Test Cases
Potency
Where to target
Who to target?
"BUG"
PDT
Potential Defect Type
Fault Coverage
"What PDTs uncovered by test cases"
Countability
"Proving sufficiency of test cases"
Conformance:Robustness
"Distribution of +ve/-ve test cases"
Levelization
"Optimal targeting"
Test case immunity
"No defect yield from test cases"
IMMUNITY
Resistant to bugs
i.e. hardened entities

Let us examine the key properties/measures of these aspects (of potency) in detail.

1. Test cases ("Drug")

Are the test cases sufficient? Assess the property of countability. What is this? If the behavior of an element under test is described as a set of conditions (i.e. a behavioral model) then an optimal combination of these conditions would result in "countable scenarios" i.e. irrespective of who designs, the number of scenarios is indeed the same. Instantiating each scenario with optimal combinations of inputs would result in test cases. The key element to note is that the number of test cases can be logically assessed for sufficiency.

The second measure is the distribution of test cases that are conformance/robustness (+/-). This is a function of #inputs and values for each input. Note that inputs can be discerned from the conditions in the behavioral model and the values for each input mechanically generated based on the input specification. Hence this measure can allow us to ascertain if our distribution is indeed good enough.

Finally segregation of test cases into quality levels (See the earlier article "Form & Structure matters in Nov 2011") enables test cases to be purposeful and enables a clear targeting of what type of defect a test case is targeting (i.e. enable the drug to reach the target area quickly).

2. Entity ("Area")

This is about examining if we are indeed covering the entire area well enough. Measuring how much external area is "Requirements coverage/traceability" whilst measuring the internal area covered is what "Code coverage" focuses on. Note that these are the typical measures that we use.

These measures describe the area that we cover, but not the types of defects we are looking for in that area. This is what we discuss next.

3. Potential Defect Type ("Fault coverage")

Tracing the test cases to the potential types of defects that they can uncover enables the test cases to be purposeful i.e. what issues am I interested in uncovering? (This is the central theme of HBT - "Hypothesis Based Testing").

So examining the properties/measures of Countability, Conformance:Robustness, Test case distribution of test cases by quality levels (Quality levels outlined in HBT cookbook http://slidesha.re/qBMNiy) and Requirements/Code/Fault coverage can allow a scientific assessment of potency of test cases.

But could the system under test be immune?

4. Immunity

The pesticide paradox ("The Art of Software Testing by Boris Bezier) states that pesticide that is used to kill a pest makes the pest resistant to the pesticide necessitating newer pesticides to be created and applied. In the context of software, this could be understood as "software has hardened" i.e. these types of bugs have been removed and probably is not present, hence test cases that are focused on uncovering these kinds of defects will not find any.

Examining the yield of test case over time will give an indication of the "resistance or immunity". Hence tracking the outcome of each test case "defect yield" (i.e. did they result in a defect) every time we run it is key to understanding this property.

The next time you design/review test cases, look at these properties to assess potency.

You want to say "I M POTENT" not "IMPOTENT".

Back To Index

Cheers.

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.

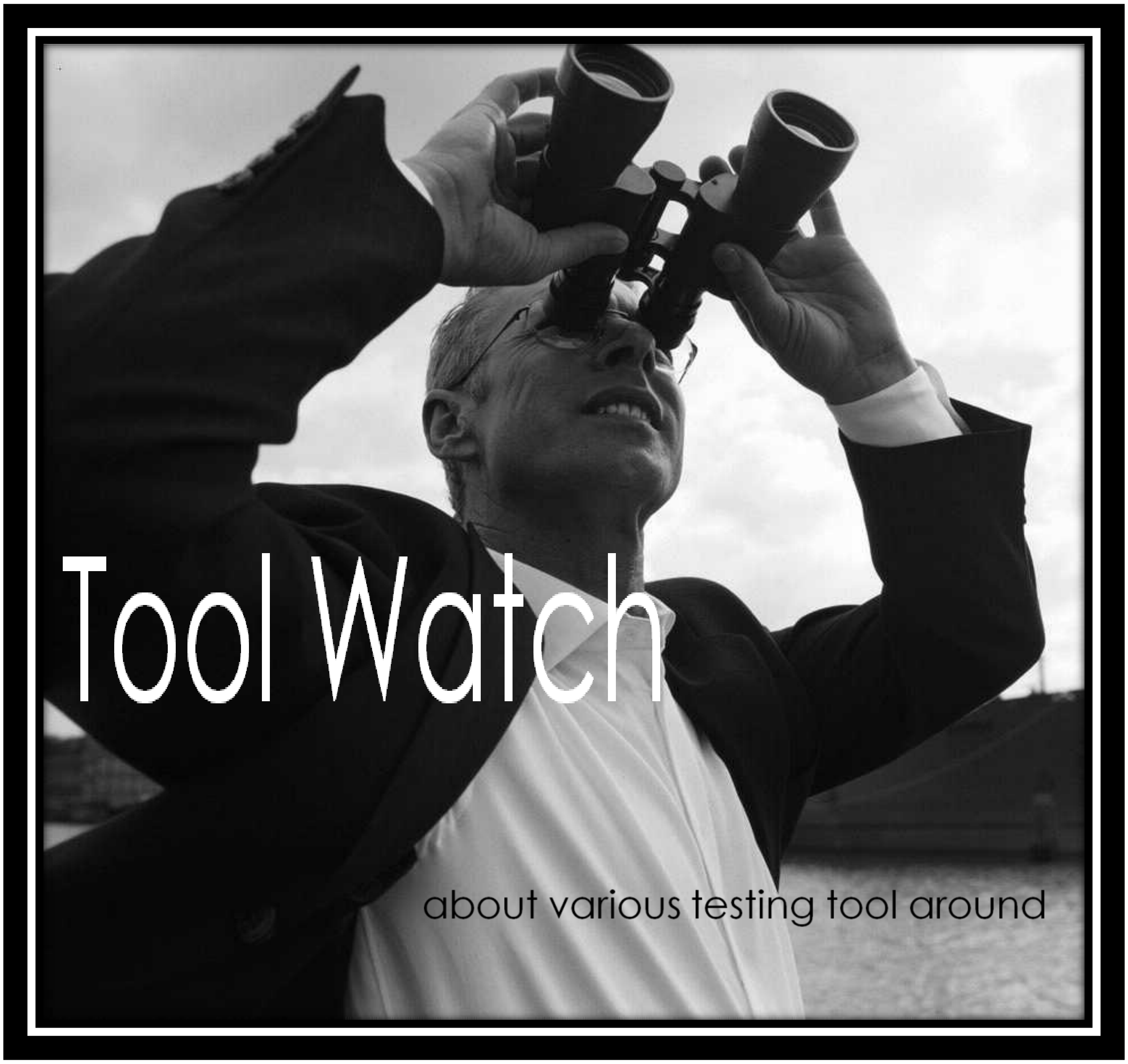
Quality Testing also provides daily Polls and sample tests for certification exams, to make tester to think, practice and get appropriate aid.

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 3

By Chris Philip

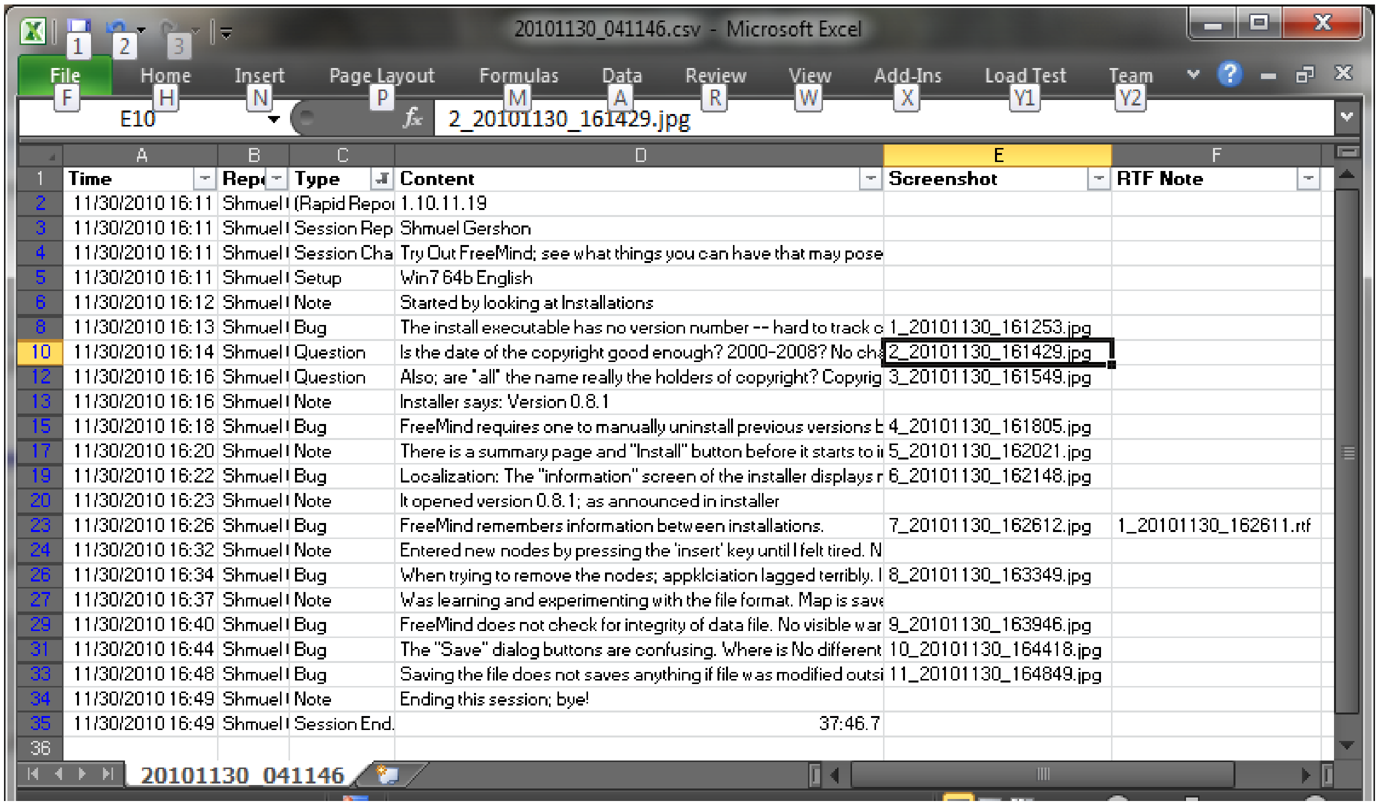
❖ View the session report

1. Find the appropriate *.CSV, by looking at the creation date or the file name.
   A session started on 21/11/2010 08:35 will be called 20101121_083548.csv, for example.

2. Open it in your favorite spreadsheet (*usually double clicking the file is enough*).

3. Now, according to the capabilities of your spreadsheet, you can re-order the notes per type, per tester, per time… You can count sessions and times and bugs and apply filters.

4. The file name of the note attachments is displayed in the same line, in a different column.



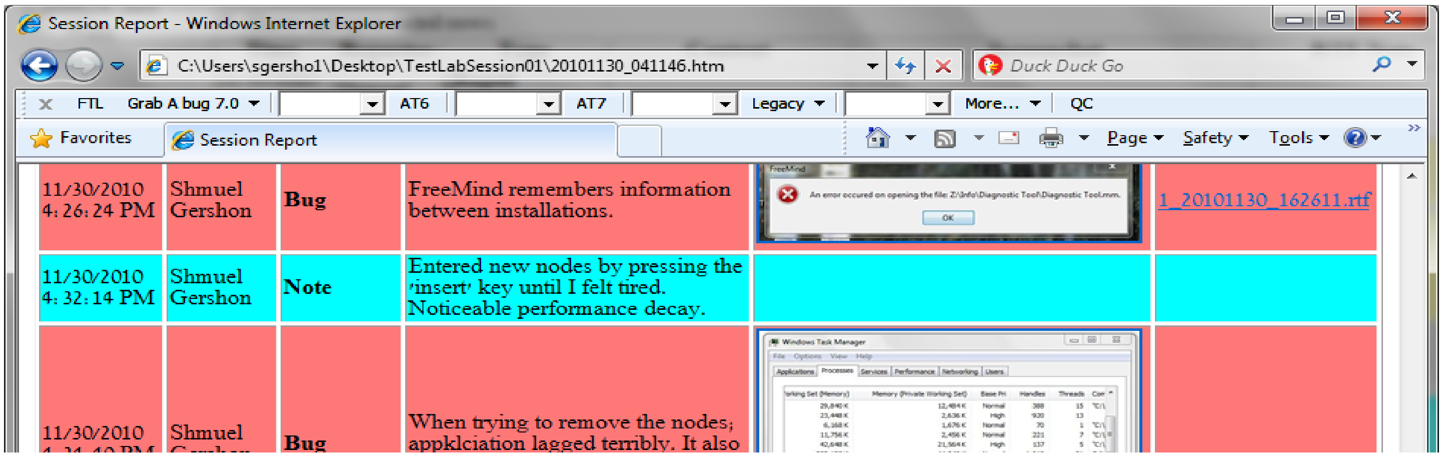
	A	B	C	D	E	F
1	Time	Rep	Type	Content	Screenshot	RTF Note
2	11/30/2010 16:11	Shmuel	(Rapid Repor	1.10.11.19		
3	11/30/2010 16:11	Shmuel	Session Rep	Shmuel Gershon		
4	11/30/2010 16:11	Shmuel	Session Cha	Try Out FreeMind; see what things you can have that may pose		
5	11/30/2010 16:11	Shmuel	Setup	Win7 64b English		
6	11/30/2010 16:12	Shmuel	Note	Started by looking at Installations		
8	11/30/2010 16:13	Shmuel	Bug	The install executable has no version number -- hard to track c	1_20101130_161253.jpg	
10	11/30/2010 16:14	Shmuel	Question	Is the date of the copyright good enough? 2000-2008? No ch	2_20101130_161429.jpg	
12	11/30/2010 16:16	Shmuel	Question	Also; are "all" the name really the holders of copyright? Copyrig	3_20101130_161549.jpg	
13	11/30/2010 16:16	Shmuel	Note	Installer says: Version 0.8.1		
15	11/30/2010 16:18	Shmuel	Bug	FreeMind requires one to manually uninstall previous versions b	4_20101130_161805.jpg	
17	11/30/2010 16:20	Shmuel	Note	There is a summary page and "Install" button before it starts to i	5_20101130_162021.jpg	
19	11/30/2010 16:22	Shmuel	Bug	Localization: The "information" screen of the installer displays r	6_20101130_162148.jpg	
20	11/30/2010 16:23	Shmuel	Note	It opened version 0.8.1; as announced in installer		
23	11/30/2010 16:26	Shmuel	Bug	FreeMind remembers information between installations.	7_20101130_162612.jpg	1_20101130_162611.rtf
24	11/30/2010 16:32	Shmuel	Note	Entered new nodes by pressing the 'insert' key until I felt tired. N		
26	11/30/2010 16:34	Shmuel	Bug	When trying to remove the nodes; appklciation lagged terribly. I	8_20101130_163349.jpg	
27	11/30/2010 16:37	Shmuel	Note	Was learning and experimenting with the file format. Map is save		
29	11/30/2010 16:40	Shmuel	Bug	FreeMind does not check for integrity of data file. No visible war	9_20101130_163946.jpg	
31	11/30/2010 16:44	Shmuel	Bug	The "Save" dialog buttons are confusing. Where is No different	10_20101130_164418.jpg	
33	11/30/2010 16:48	Shmuel	Bug	Saving the file does not saves anything if file was modified outsi	11_20101130_164849.jpg	
34	11/30/2010 16:49	Shmuel	Note	Ending this session; bye!		
35	11/30/2010 16:49	Shmuel	Session End.	37:46.7		
36						

❖ View the session report in HTML

The HTML report will allow you to see the screenshot attachments and the extended notes links embedded with the session notes. Additionally, it will allow you to change the visuals of the report to include the logo of your company or project, for example. These two benefits make it great for discussing the session with managers.

To create a report of a session *.CSV file (*either of single session or cumulative sessions*), use the following command in command line:

```
C:\> RapidReporter.exe –tohtml <nameofsessionfile.csv>
```


Session Report - Windows Internet Explorer
C:\Users\sgershо1\Desktop\TestLabSession01\20101130_041146.htm
Duck Duck Go
FTL
Grab A bug 7.0
AT6
AT7
Legacy
More...
QC
Favorites
Session Report
Page
Safety
Tools

11/30/2010 4:26:24 PM	Shmuel Gershon	**Bug**	FreeMind remembers information between installations.		1_20101130_162611.rtf
11/30/2010 4:32:14 PM	Shmuel Gershon	**Note**	Entered new nodes by pressing the 'insert' key until I felt tired. Noticeable performance decay.		
11/30/2010	Shmuel	**Bug**	When trying to remove the nodes; appklciation lagged terribly. It also		

❖ Change the visuals of the HTML report

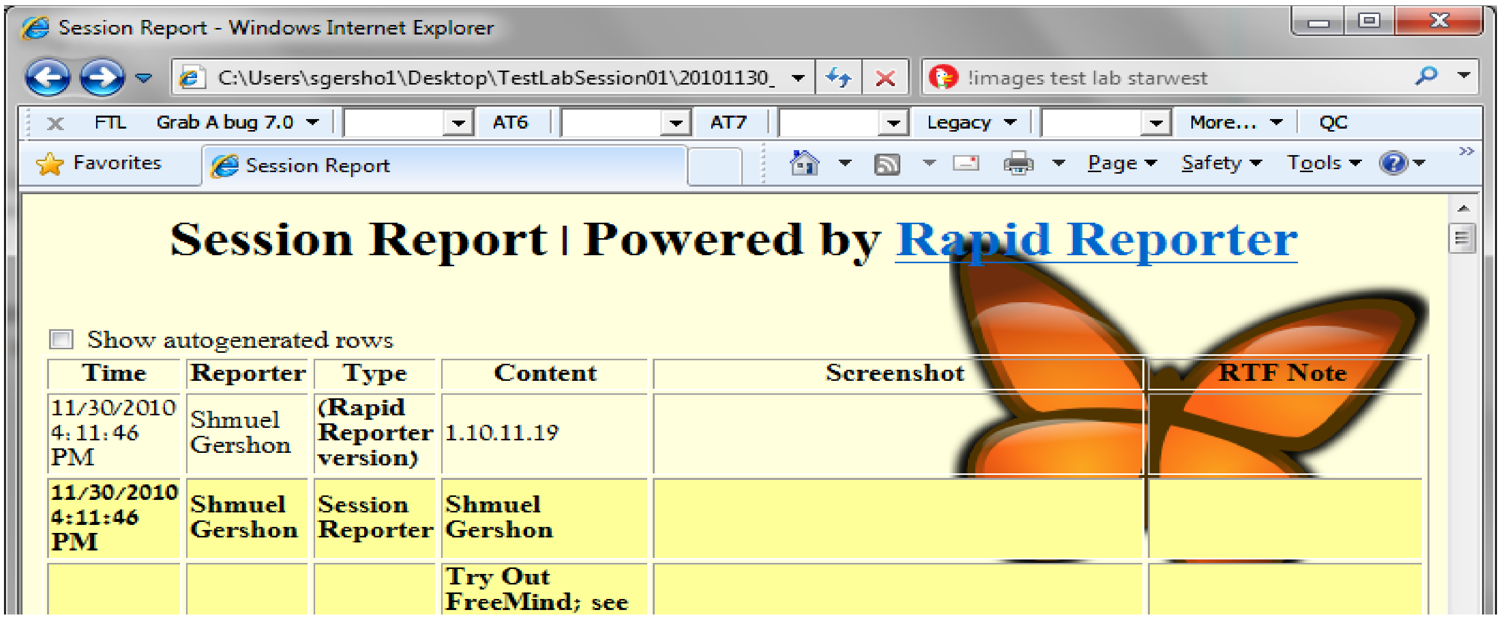
Changing the visuals of the HTML report is useful when you want to add your company's logo to the report, when you want the colors of the report to match the ones in other reports you provide, when you prefer to hide or emphasize one type of note… Almost all visuals of a report can be manipulated.

1. Use a file named style.css in the same folder as the HTML report.

2. You will need acquaintance with CSS to customize the report. Looking at the HTML source should elucidate what to edit according to your preferences.

An example of stylesheet would be:

```
body
{
background: #FFFFDD;
}
#allbody
{
background-image:url('http://upload.wikimedia.org/wikipedia/commons/d/d9/Free_Mind.png');
background-repeat:no-repeat;
background-position:right top;
}
table tr.Bug {font-weight: bold;}
H1 {color:#000000;}
```



Session Report - Windows Internet Explorer
C:\Users\sgersho1\Desktop\TestLabSession01\20101130_
!images test lab starwest
FTL
Grab A bug 7.0
AT6
AT7
Legacy
More...
QC
Favorites
Session Report
Page
Safety
Tools

Session Report | Powered by Rapid Reporter

Show autogenerated rows

Time	Reporter	Type	Content	Screenshot	RTF Note
11/30/2010 4:11:46 PM	Shmuel Gershon	(Rapid Reporter version)	1.10.11.19		
11/30/2010 4:11:46 PM	**Shmuel Gershon**	**Session Reporter**	**Shmuel Gershon**		
			Try Out FreeMind; see		

to be continued in next issue...

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 QUALITY TESTING

Testing PUZZLES by Sebi

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

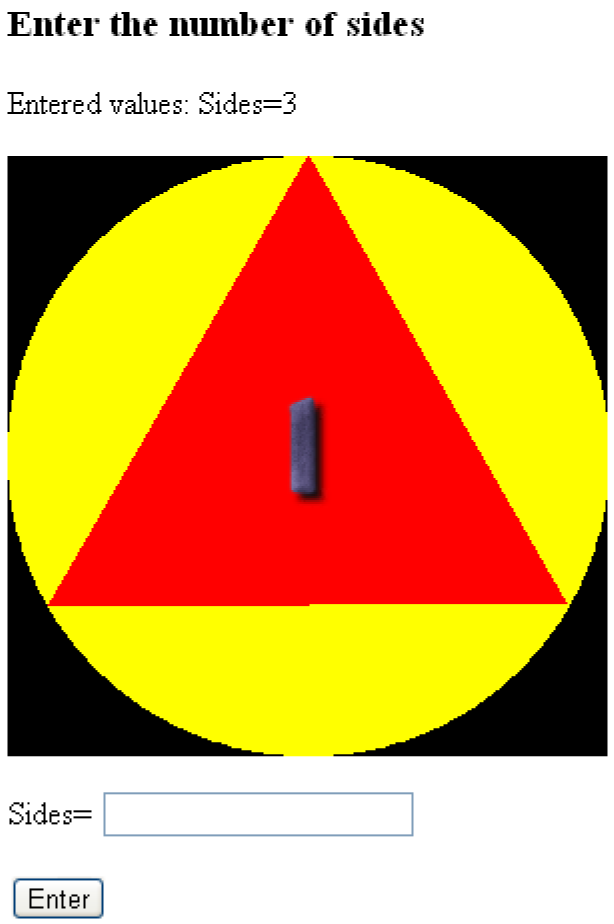
Send -> teatimewithtesters@gmail.com with Subject: Testing Puzzle

Gear up guys.......

It's Time To Tease your Testing Bone

Puzzle "Play with the Patterns"

"For a specific range of inputs in this application http://testalways.com/3/ there is an image generated and a number is shown in the middle of it. For bigger inputs there is no image generated though. If the same patterns would be respected and the image generated (like for smaller inputs), what would be the number in the middle when using as input 1000000?"



Biography

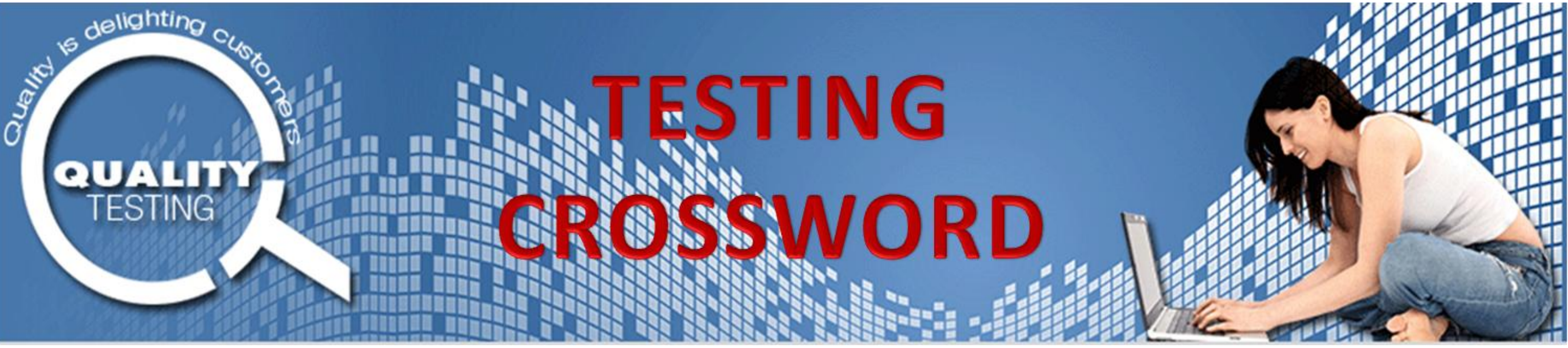
Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years. He is currently hosting European Weekend Testing.

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

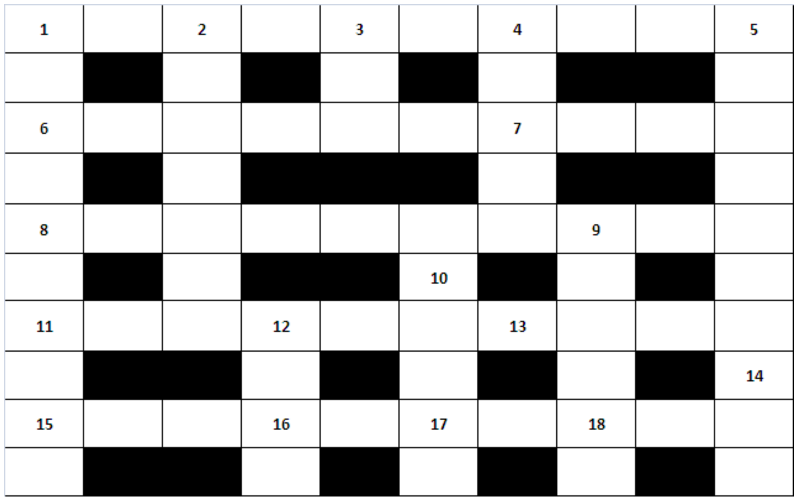
He tweets as @testalways.

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

Back To Index



TESTING CROSSWORD



Horizontal:

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

6. A black box test design technique in which test cases are designed based upon the definition of the input domain and/or output domain is called _____ testing (6)

7. A tool that facilitates the recording and status tracking of incidents, in short form (3)

8. Testing of individual components in isolation from surrounding components, with surrounding components being simulated by stubs and drivers, if needed is called ____ testing (9)

11. The ____ automation platform is an open, extensible, and neutral system for .NET that provides a common object model, runtime services and tools that may be leveraged by any number of test frameworks (6)

13. Testing in which all paths in the program source code are tested at least once, is called _______ testing (4)

15. The short form of Observation Driven Testing (3)

16. It is a free command line tool for generating stressful text data for field testing as well as Random testing (2)

17. Checks for memory leaks or other problems that may occur with prolonged execution, is called _____ testing. In short form (2)

18. It is a Selenium 1 (Selenium RC) client library that provides a programming interface (API), i.e., a set of functions, which run Selenium commands. The first word (3)

Vertical:

1. It is a tool provide easy cross browser testing with Selenium in the cloud (10)

2. ________ testing technique can be used for testing non-functional attributes such as reliability and performance (6)

3. It is a testing aimed at showing software does not work and also known as "test to fail", in short form (2)

4. Testing of individual software components is called ____ testing (4)

5. Testing by means of a random selection from a large range of inputs and by randomly pushing buttons, ignorant on how the product is being used. Is called _____ testing (6)

9. A mechanism to produce the expected outcomes to compare with the expected outcomes of the Software Under Test (6)

10. An environment for system or user acceptance testing which is as close to field use as possible. It is called _____ office (5)

12. A white box testing technique that exercises program loops, it is called _______ testing (4)

14. The tool developed to ease the issues encountered by having to perform Quality Assurance tests across a variety of hardware and software combinations (3)

Answers for Last Month's Crossword:

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

Answer for the last puzzle : **980993971**

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

Learned about your magazine from one testing conference.

I was curious to read your magazine when I heard people praising you people.

Good job.

- Nikol S.

My Voice on "Teach-testing"

Yes software testing should be taught as a separate course in universities. This will ensure giving appropriate and in depth knowledge about testing stream and at the same time prevent private institutes bullying in software testing training field and loss of student's money and time. Freshers will not get deceived by private institutes for paying huge amount of money for very less knowledge about software testing.

I will suggest some kind of aptitude test also to find student's aptitude as well as liking towards software testing.

- Prajakta Kanade

There are two things for which I eagerly wait every month.

First one is my salary and second is Tea-time with Testers ! ☺

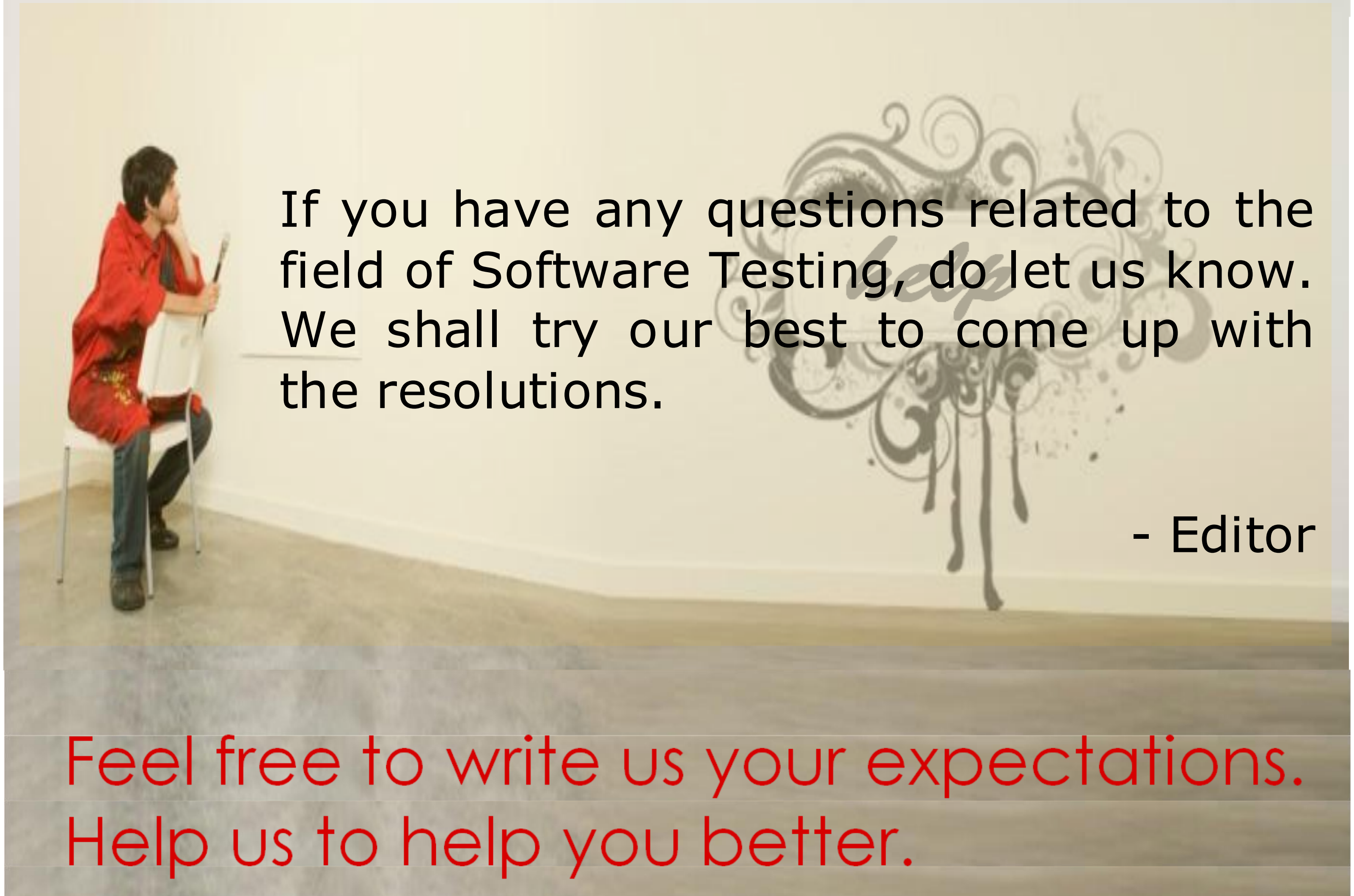
- Sudha Ramalingam

Thank you for creating this magazine. Your articles have helped me on multiple occasions. Thanks again.

- Kirtee Damle

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 nexxt 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)

Romil Gupta (Pune, India)

Eusebiu

Romil

Tech -Team:

Subhodip Biswas (Mumbai, India)
Chris Philip (Mumbai, India)
Gautam Das (Mumbai, India)

Subhodip

Chris

Gautam

|| 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