

Tea-time with Testers

6th Anniversary Issue

Articles by –

Jerry Weinberg

John Stevenson

T Ashok

Juan Salinas


Viktor Slavchev

Baris Sarialioglu

Joel Montvelisky



Over a Cup of Tea with Anne-Marie Charrett



MARK YOUR CALENDARS



Dive Deeper
into Developing
and Testing
Software for
Mobile and The
Internet of Things



Mobile Dev + Test

A TECHWELL EVENT

IoT Dev + Test

A TECHWELL EVENT

April 24-28, 2017

Westin San Diego | San Diego, CA

<https://well.tc/wory>



TEA-TIME WITH TESTERS

First Indian testing magazine to reach 115 countries in the world !

Created and Published by:

Tea-time with Testers.

B2-101, Atlanta, Wakad Road

Pune-411057

Maharashtra, India.

Editorial and Advertising Enquiries:

- editor@teatimewithtesters.com
- sales@teatimewithtesters.com

Lalit: (+91) 15227220745

Pratik: (+91) 15215673149

This ezine is edited, designed and published by **Tea-time with Testers**. No part of this magazine may be reproduced, transmitted, distributed or copied without prior written permission of original authors of respective articles.

Opinions expressed or claims made by advertisers in this ezine do not necessarily reflect those of the editors of **Tea-time with Testers**.

Editorial

Looking back and forward

Dear reader

First of all, I would like to thank you all for your great response to State of Testing survey 2017. Over 1500 testers have filled it this year and we hope more and more testers will join us in this journey to make the report even more useful.

Well, 2016 had been an eventful year for Tea-time with Testers family. While we did great on certain initiatives like the survey itself, then Techno-talks with Lalit show on TV for Testers, we got slowed down on publishing TTWT issues on monthly basis. I do feel bit bad about it but on other hand I am happy that we are serving the community in more than just one way, with everything on non-commercial level.

Around this same period 6 years back, we started “Tea-time with Testers” because we felt the need to give community something back. I am glad that we have reached till here, are still able to do that and with more mediums than before. I always count my individual contribution to the community (be it about teaching AST’s BBST Foundations, or assisting James Bach in RSTA, showing up and presenting at conferences, taking free public workshops or writing my own blogs) separate from TTWT work, but I do feel that it’s TTWT itself that has enabled me to make contributions on personal level too. And this goes without saying dear readers, that this all could happen only because of your love and support. A big thank you once again for being there and encouraging us.

Indeed, all these things require a great deal of time, commitment and passion and hence I would also like to thank our team of editors, advisors, tech team and online co-ordination team for showing such strong commitment and passion. And having said that we have now more than one activity to look after, we also need more and more passionate testers to help us taking this work further. If you think you have what it takes to be part of our awesome team, I welcome you to join us (age, sex and location – no bar!).

I am excited about stepping into 7th year of community service and look forward to serve you for many years to come.

Enjoy reading our anniversary special edition. And we’ll meet again soon!

Sincerely Yours,



- **Lalitkumar Bhamare**

editor@teatimewithtesters.com

@Lalitbhamare / @TtimewidTesters



QuickLook



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

Testing Skills- part 7- 26

Should we get rid of all Blackbox testers? 22

In the School of Testing

Improving Test processes by improving test data – 32

Outdated Testing Concepts #2 – 35

Complexity is also a Bug – 44

T' Talks

Practice (not process) makes it perfect, rapidly- 41

Over a Cup of Tea with Anne-Marie Charrett

Family de Tea-time with Testers



What's making News?

Mobile Dev + Test and IoT Dev + Test

April 24-28, 2017 San Diego, CA

TechWell, a leader in software development lifecycle news and expert articles, puts on one of the most in-depth learning conferences on mobile and IoT testing in the industry. Explore how to improve specific skills like using Gradle, Appium®, Angular JS, Swift, and more. Or expand your knowledge of mobile test automation, app security testing, developing your mobile test and quality plans, and so much more.

The hands-on workshop-like tutorials are consistently the most popular sessions attended at the conference. With full-day and half-day tutorials on skills you need to improve or master your software testing skills for mobile and IoT projects, Mobile Dev + Test and IoT Dev + Test are the conferences for you.

The collocated Mobile Dev + Test and IoT Dev + Test conferences have consistently received 95% approval rating from attendees with 95% of attendees wanting to return to learn new tips, techniques and methods each year.

Mobile Dev + Test
IoT Dev + Test 
A TECHWELL EVENT

Register using
promo code
MDCM and
save up to \$200

APRIL 24-28, 2017
San Diego, CA

See what past attendees liked most about the conferences:

"I learned a ton, met great people, great presentations in expo of available products and solutions we could use. Exciting" Chris Mosconi, QA Lead, Under Armour Connected Fitness

"Tutorials and sessions were great, the tables at lunch were great, and having it in San-Diego was perfect!" Sandra Nagel, Senior Quality Assurance Engineer, CoStar Group

Additionally there will be an Expo hall full of the latest and greatest software solutions and services there to answer your questions and help you solve your biggest mobile and iot testing or development problems. Sponsors such as Apica, TestPlant, Mobile Labs, Sauce Labs, Testilo, Google and many more will be there to meet and explore solutions with.

Register for the Mobile Dev + Test conference and receive full access to the IoT Dev + Test conference. Use promo code MDCM and save up to \$200 off, plus when you register by March 24 you can save up to an additional \$200 with early bird pricing that is a combined savings of up to \$400!

[Learn more](#)

8th INTERNATIONAL TESTISTANBUL CONFERENCE

AGILE TESTING

April 25th, 2017

Renaissance Istanbul Polat Bosphorus Hotel

Turkish/English
simultaneous
translation

Keynote Speakers



**Geoffrey John
Thompson**

*Consultancy Director
Experimentus*



**Jan Jaap
Cannegieter**

*Principal Consultant
Squerist*



**Michael
Bolton**

*Principal
DevelopSense*



**Philip
Lew**

*CEO
XBOsoft*

EARLY BIRD
290 USD + VAT

Until January 27th, 2017

REGULAR FEE
360 USD +VAT

**10% discount for ISTQB certificate owners*

Event Partners

keytorc

Exhibitors

andagon



Supporters



Organized by



Register now: testistanbul.org

INTERVIEW

Anne-Marie Charrett has been a familiar name to me since my Context Driven Testing awakening happened. If I remember correctly, we first talked back in 2011 while she wrote some amazing articles for Tea-time with Testers. Her “Career path for Testers” article is still fresh in my mind. And from there I kept reading her blogs, learned from her work.

Without a doubt, Anne-Marie is great at what she does. And she does not really need any special introduction to testers who are active in the community. If you don't know her, I suggest you to start following her [work](#).

Interviewing Anne-Marie was on my wish-list for a while now and I'm glad that it's finally happening. We do exchange testing thoughts once in a while (we're peer advisors on James Bach's RSTA) but I feel what we (me and Dirk) discussed during her interview is special in many ways. It's no wonder she is known as the 'Maverick Tester'.

Read and enjoy!

- Lalitkumar Bhamare



Over A Cup of Tea with Anne-Marie Charrett

It's an honor interviewing you, Anne-Marie. Thanks for talking with us today.

Thanks Lalit, there's nothing like a chat between friends over a nice cup of tea!

You were one of the nominees for Software Test Luminary award for 2016. How do you feel about your journey so far?

I was surprised to be nominated. My ego was definitely flattered especially when I discovered I'd been nominated alongside Elizabeth Hendrickson, someone who I respect and admire.

My testing journey is pretty much how I expected it to be. Full of ups and downs but so far pretty true to what I aspire and aim for. That is, to be true to myself, keep learning new approaches and ideas, and do great work.

Please tell us about some special moments in your testing career. Are there any interesting stories we don't know yet?

I am grateful to many people in my software testing career. In particular, those who gave me a deeper & nuanced understanding of testing is and can be. People such as Cem Kaner, Scott Barber and James Bach gave me insights into continuous enquiry and learning in the testing space. I am and will always be grateful for that.

A recent 'moment' has been to realise that 'Anne-Marie the human', is far more important than 'Anne-Marie the tester'. I am proud to have realised this. Putting my laptop aside and spending time with my family and friends used to be scary for me. Being available online 'always' is a blessing and a curse. The fear of missing 'out' for me has been a curse.

I've also learned that I don't have to respond to every request for help. Saying no, is equally as important as saying yes. Doing this, has given me renewed self-confidence and deeper connections with those I respect and admire.

What new things are you currently working on?

Right now, I feel it's important to me to immerse myself in new technologies and practices such as Continuous Delivery and the DevOps movement. I've had some amazing opportunities present themselves to me in both the FinTech & larger enterprise organizations. I'm enjoying learning new things, different mindsets. In particular concepts that challenge how I think about testing. It has given me a fresh perspective on Quality in general. I have a better appreciation of what it takes to work as a tester in this context.

How was your experience with teaching testing at the University of Technology, Sydney? How easy/difficult is it as compared to teaching professional testers?

Teaching at University was a great experience for me mostly because I was able to create my own content. I am immensely grateful to Tom McBride from UTS for giving me the autonomy to do that. I enjoyed teaching the students and still see some of them (one of them even worked for me).

The biggest difference between teaching professional testers and University students is assessment and grading. Assessment is an opportunity for feedback. Offering feedback to students is vital, but it's time consuming. I would have liked more time to offer feedback, as I believe that is where some core learning can be obtained.

You are an active founding member of “Speak-easy”. It’s an impressive initiative that we admire. How has been the journey of Speak-easy so far? Is there anything you would like to say our readers, on behalf of Speak-easy?

They say that a community should be measured by its weakest link. Nah, I just made that up! Really though, it’s a testament to our community that this initiative has gained such support. SpeakEasy was founded to mentor and encourage more women speak at testing conferences. We partner with conferences who wish to allocate a speaking slot to a new upcoming speaker. SpeakEasy then matches testers wanting to speak at these conferences with mentors willing to help them.

However, there is quite a bit of work involved matching testers and mentors and working with conferences. Both I and Fiona, are struggling to maintain this work. We are seeking volunteers willing to help us review and rework how we match mentors, and how we can provide more support to mentors.

Please recommend us some names from Speak-easy that you would like us to approach for articles and interviews.

I would really like to hear about experiences of some of our mentors. A recent speaker has been **Mirjana Kolarov** who gave an excellent talk at Eurostar this year on the topic of performance testing. Recently, James Thomas wrote an honest and courageous post on his experience speaking. It’s experiences like this that make such an initiative worthwhile.

Your blog around leaving testing to the experts generated interesting discussion in community. Do you feel that ‘independence of testers’ is still the problem with our industry? What would you advise testers/organizations for not falling into that trap?

The blog post came from a frustration of working in a field that struggles to gain respect within our industry. And, why does it feel like this is getting worse rather than better? Many push the idea that tester’s need to earn credibility and respect. I like that idea. But isn’t that a universal truth for any person working in software delivery? Why the need to highlight testers?

Here’s what I think. Yes, merit must be earned, but it’s also incumbent on the rest of a team to provide a cultures that testers can sit comfortably in. No tester should have to feel they need to be an all dancing/singing rockstar in order to prove their worth.

It’s sad, but sometimes I see tester’s needing to work twice as hard to gain credibility that many who ‘code’ would automatically be given. If we truly believe in & want cross-functional teams, there needs a willingness to accept & a respect in the diversity of mindsets. To me, anything less is lip service. In some way that blog post was a call to action to redress that balance.

You are known to be an advocate of a whole team approach to Quality. Could you tell our readers in brief what do you mean by that?

Typically, in many enterprise organisations, it’s the tester’s responsibility to ‘ensure quality’. But how can a tester ensure quality when all you ask them to do is observe how the product behaves? They don’t control budget, time, people...really, is that fair?

Team ownership of Quality means the team, not only the tester, understands what Quality means in their context and all work to that goal.

The phrase “Quality is a team responsibility” is a call to recognise that every member of a cross functional team plays a part in understanding and building quality. Product Owners aspire to build the right thing, Developers to build it right, Ops to keep the lights on in production and Testers to shine light on possible problems. Hey! What can go wrong? :)

More and more companies appear to be moving away from Quality Assurance departments and are embedding testers into agile teams. What risks/benefits do you see in this approach?

Cross Functional Teams where a tester sits with developers, product owners, UX designers and Operations can bring great benefit to designing and building and maintaining product. The biggest return I see is the ability to collaborate closely on a small piece of work. For a tester who trades in information this is priceless. We get to see and understand reasons why decisions on design and development are made. We see the challenges in deployment and maintenance.

However, mixing people with diverse skillsets and mindsets means significant effort is required to create a culture where everyone is offered respect and inclusion. Unfortunately, sometimes I see tester's being 'of' the team but not 'in' the team. I think that's sad and better effort is required to understand how diversity of skills can add value to a team.

At present, what are the biggest challenges you think the testing community has to solve?

Context Driven Testing came from a bunch of practitioners who got sick of listening to non-practitioners dictate how testing should be performed using 'best practice'. It would do no harm for our community to go back this principle.

Let's look and learn from those who are performing testing. In the context of cross functional teams, & continuous delivery that may means testers, developers, Business Analysts, UX and Ops. In this context, all these specialists to some degree test. I'm interested in learning how we can apply our testing expertise to this new context. For example, how given a Continuous Deployment pipeline, where is the real risk and how as testers can we help mitigate that risk through tangible evidence and data?

I'd also like to see greater respect for diversity of views, even if they differ to our own. There's an ideology in Context Driven Testing that schools are paradigms making it pointless to attempt to discuss or debate. Maybe it's the tester in me, but I struggle with that. As a tester, I want/need to experiment to investigate if that is true. To not, restricts my learnings and I feel stifled. If that happens, then I will go somewhere else where I can investigate and learn.

Seriously, never try and hold a good tester down! :)

For the first time CAST is happening outside of North America, and it's taking place in Sydney, Australia. What makes CASTx17 different?

Well, for starters, as you pointed out, it is taking place outside of North America. This has been something many AST members have been requesting for a while. It is great to see the AST listening to and acting on what the community wants.

For a while now, I have wanted to create a program that invites ideas from the people committed to delivering quality products. People who perform testing as a part of their daily efforts but don't identify themselves as testers. I think these people have a lot to say that we testers could find useful, and I'm hoping they can find something useful from what we have to share.

CAST always has a diversity of ideas and I think this conference is no different in that regard. We have talks on Usability, Security, Performance, Auditing, Automation, Tooling and Exploratory Testing just to mention a few.

I am hoping for a conference where we can share, listen and discuss ideas in a respectful and safe environment.

You have written amazing articles on multiple occasions for Tea-time with Testers in past. What would you recommend us for our further improvement?

How about creating magazines of existing material that are themed collections? For example, 2017 collection of interviews, or articles on a certain topic?

Tea & Testing

with

Jerry Weinberg



What Is Quality? Why Is It Important?

"You can fool all of the people some of the time, and some of the people all of the time, but you can't fool all of the people all of the time."- Abraham Lincoln

People in the software business put great stress on removing ambiguity, and so do writers. But sometimes writers are intentionally ambiguous, as in the title of this book.

"Quality Software Management" means both "the management of quality software" and "quality management in the software business," because I believe that the two are inseparable. Both meanings turn on the word "quality," so if we are to keep the ambiguity within reasonable bounds, we first need to address the meaning of that often misunderstood term.

1.1 A Tale Of Software Quality

My sister's daughter, Terra, is the only one in the family who has followed Uncle Jerry in the writer's trade. She writes fascinating books on the history of medicine, and I follow each one's progress as if it were one of my own. For that reason, I was terribly distressed when her first book, *Disease in the Popular American Press*, came out with a number of gross typographical errors in which whole segments of text disappeared (see Figure 1-1). I was even more distressed to discover that those errors were caused by an error in the word processing software she used—CozyWrite, published by one of my clients, the MiniCozy Software Company.

The next day, too, the Times printed a letter from "Medicus," objecting to the misleading implication in the microbe story that diphtheria could ever be inoculated against; the writer flatly asserted that there would never be a vaccine for this disease because, unlike smallpox, diphtheria re-

Because Times articles never included proof—never told how people knew what they claimed—the uninformed reader had no way to distinguish one claim from another.

Figure 1-1. Part of a sample page from Terra Ziporyn's book showing how the CozyWrite word processor lost text after "re-" in Terra's book.

Terra asked me to discuss the matter with MiniCozy on my next visit. I located the project manager for CozyWrite, and he acknowledged the existence of the error.

"It's a rare bug," he said.

"I wouldn't say so," I countered. "I found over twenty-five instances in her book."

"But it would only happen in a book-sized project. Out of over 100,000 customers, we probably didn't have 10 who undertook a project of that size as a single file."

"But my niece noticed. It was her first book, and she was devastated."

"Naturally I'm sorry for her, but it wouldn't have made any sense for us to try to fix the bug for 10 customers."

"Why not? You advertise that CozyWrite handles book-sized projects."

"We tried to do that, but the features didn't work. Eventually, we'll probably fix them, but for now, chances are we would introduce a worse bug—one that would affect hundreds or thousands of customers. I believe we did the right thing."

As I listened to this project manager, I found myself caught in an emotional trap. As software consultant to MiniCozy, I had to agree, but as uncle to an author, I was violently opposed to his line of reasoning. If someone at that moment had asked me, "Is CozyWrite a quality product?" I would have been tongue-tied.

How would you have answered?

1.2 The Relativity of Quality

The reason for my dilemma lies in the *relativity of quality*. As the MiniCozy story crisply illustrates, what is adequate quality to one person may be inadequate quality to another.

1.2.1. Finding the relativity

If you examine various definitions of quality, you will always find this relativity. You may have to examine with care, though, for the relativity is often hidden, or at best, implicit.

Take for example Crosby's definition:

"Quality is meeting requirements."

Unless your requirements come directly from heaven (as some developers seem to think), a more precise statement would be: "Quality is meeting some person's requirements."

For each different person, the same product will generally have different "quality," as in the case of my niece's word processor. My MiniCozy dilemma is resolved once I recognize that

- a. To Terra, the people involved were her readers.
- b. To MiniCozy's project manager, the people involved were (the majority of) his customers.

1.2.2 Who was that masked man?

In short, quality does not exist in a non-human vacuum.

Every statement about quality is a statement about some person(s).

That statement may be explicit or implicit. Most often, the "who" is implicit, and statements about quality sound like something Moses brought down from Mount Sinai on a stone tablet. That's why so many discussions of software quality are unproductive: It's my stone tablet versus your Golden Calf.

When we encompass the relativity of quality, we have a tool to make those discussions more fruitful. Each time somebody asserts a definition of software quality, we simply ask, "Who is the person behind that statement about quality."

Using this heuristic, let's consider a few familiar but often conflicting ideas about what constitutes software quality:

- a. "Zero defects is high quality."
 - 1. to a user such as a surgeon whose work would be disturbed by those defects
 - 2. to a manager who would be criticized for those defects
- b. "Lots of features is high quality."
 - 1. to users whose work can use those features—if they know about them
 - 2. to marketers who believe that features sell products
- c. "Elegant coding is high quality."
 - 1. to developers who place a high value on the opinions of their peers
 - 2. to professors of computer science who enjoy elegance
- d. "High performance is high quality."
 - 1. to users whose work taxes the capacity of their machines
 - 2. to salespeople who have to submit their products to benchmarks
- e. "Low development cost is high quality."
 - 1. to customers who wish to buy thousands of copies of the software
 - 2. to project managers who are on tight budgets

f. "Rapid development is high quality."

1. to users whose work is waiting for the software
2. to marketers who want to colonize a market before the competitors can get in

g. "User-friendliness is high quality."

1. to users who spend 8 hours a day sitting in front of a screen using the software
2. to users who can't remember interface details from one use to the next

1.2.3 The political dilemma

Recognizing the relativity of quality often resolves the *semantic* dilemma. At the start of a book on quality, this is a monumental contribution, but it still does not resolve the *political* dilemma:

More quality for one person may mean less quality for another. For instance, if our goal were "total quality," we'd have to do a summation over *all* relevant people. Thus, this "total quality" effort would have to *start* with a comprehensive requirements process that identifies and involves all relevant people. Then, for each design, for each software engineering approach, we would have to assign a quality measure for each person. Summing these measures would then yield the total quality for each different approach.

In practice, of course, no software development project ever uses such an elaborate process. Instead, most people are eliminated by a prior process that decides:

Whose opinion of quality is to count when making decisions?

For instance, the project manager at MiniCozy decided, without hearing arguments from Terra, that her opinion carried minuscule weight in his "software engineering" decision. From this case, we see that software engineering is *not* a democratic business.

Nor, unfortunately, is it a *rational* business, for these decisions about "who counts" are generally made on an emotional basis.

1.3 Quality Is Value To Some Person

The political/emotional dimension of quality is made evident by a somewhat different definition of quality. The idea of "requirements" is a bit too innocent to be useful in this early stage, because it says nothing about *whose* requirements count the most. A more workable definition would be this:

"Quality is value to some person."

By "value," I mean, "What are people willing to pay (do) to have their requirements met." Suppose, for instance, that Terra were not my niece, but the niece of the president of the MiniCozy Software Company. Knowing MiniCozy's president's reputation for impulsive emotional action, the project manager might have defined "quality" of the word processor differently. In that case, Terra's opinion would have been given high weight in the decision about which faults to repair.

In short, the definition of "quality" is always political and emotional, because it always involves a series of decisions about whose opinions count, and how much they count relative to one another. Of course,

much of the time these political/emotional decisions—like all important political/emotional decisions—are hidden from public view.

Most of us software people like to appear rational. That's why very few people appreciate the power of this definition of quality. Hopefully, its power will be revealed throughout these volumes.

What makes our task even more difficult is that most of the time these decisions are hidden even from the conscious minds of the persons who make them. That's why one of the most important actions of a quality manager is to bring such decisions into consciousness, if not always into public awareness. That will be one of our major tasks.

To be continued...

Speakers at Global Testing Retreat 2017 #ATAGTR2017
16th-17th March, 2017 - Pune

Mukta Aphale

Masa Maeda

Ramit Kaul

Gina Enache

Schalk Cronje

George Dinwiddie

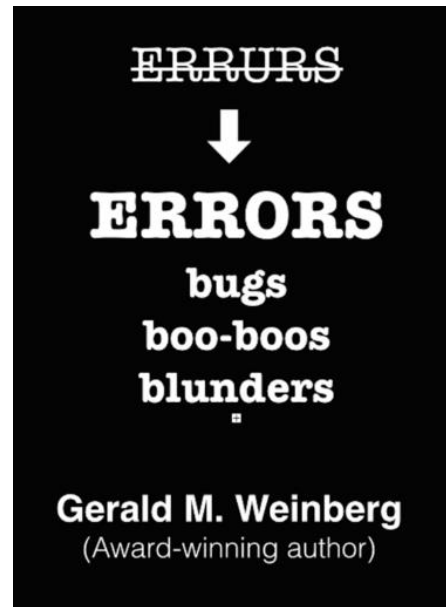
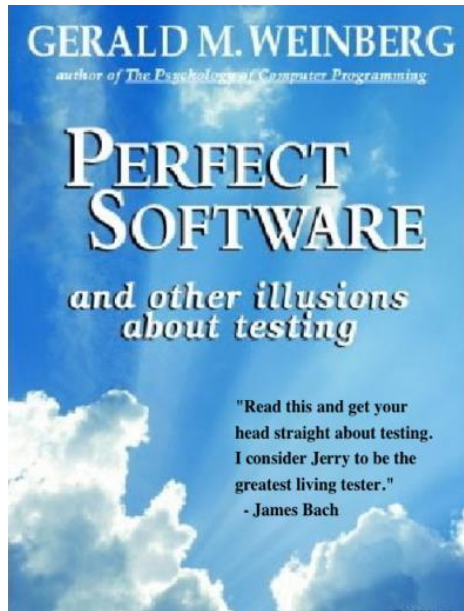
Your Name

Do you **ALSO want to be a **SPEAKER** at **GTR 2017** ?**

Call for Papers #ATAGTR2017 is now ON!
gtr.agiletestingalliance.org/papers.html



If you want to learn more about testing and quality, take a look at some of Jerry's books, such as:



People Skills—Soft but Difficult



Curious to know why you should get 'People Skills' bundle by Jerry? [Then check this out](#)

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 **SoftwareTest 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](#)

Jerry Weinberg has been observing software development for more than 50 years.

Lately, he's been observing the Agile movement, and he's offering an evolving set of impressions of where it came from, where it is now, and where it's going. If you are doing things Agile way or are curious about it, **this book** is for you.

Know more about Jerry's writing on software on **his website**.

AGILE IMPRESSIONS



Gerald M. Weinberg

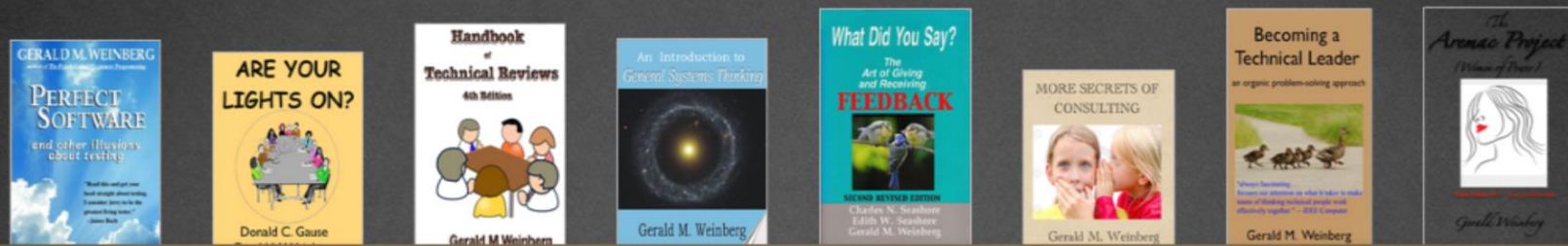
TTWT Rating: ★★★★★

The Bundle of Bliss

Buy Jerry Weinberg's all testing related books in one bundle and at unbelievable price!

The Tester's Library

Sold separately, these books have a minimum price of \$83.92 and a suggested price of \$83.92...



The suggested bundle price is **\$49.99**, and the minimum bundle price is...

\$49.99!

Buy the bundle now!

The Tester's Library consists of eight five-star books that every software tester should read and re-read. As bound books, this collection would cost over \$200. Even as e-books, their price would exceed \$80, but in this bundle, their cost is only \$49.99.

The 8 books are as follows:

- Perfect Software
- Are Your Lights On?
- Handbook of Technical Reviews (4th ed.)
- An Introduction to General Systems Thinking
- What Did You Say? The Art of Giving and Receiving Feedback
- More Secrets of Consulting
- Becoming a Technical Leader
- The Aremac Project

Know more about this bundle

A photograph of a green, conical pendulum bob hanging from a thin wire. The bob is positioned directly above a circular, swirling pattern etched into a surface of light-colored sand. The entire scene is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk

Should we get rid of all our black box testers?

- by Juan Salinas

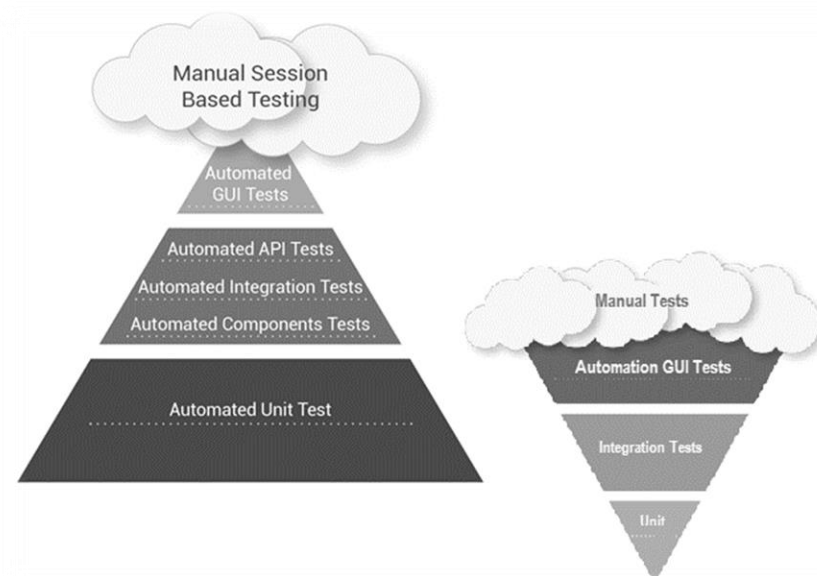


Should we get rid of our all black-box testers and turn them into automators?

No, and no. and here is why I think so:

It is evident now that if we want to move at the pace of continuous deployment (or any other level of the continuous development pipeline) we need to embrace automation and DevOps as being a key, must-have activity, in our engineering teams. The well-known pyramid approach (not the ice-cream cone which is now considered an anti-pattern) is spot on to this discussion.

Here's an illustration I liked of these approaches:



Let's analyze the pyramid approach piece by piece.

The foundation of it all is unit testing, and it sure should be since it supports all of the rest. It's mainly made up of robust unit tests at the lowest level of implementation. They provide the quickest feedback and developers can fix problems quite rapidly.

I think the development team should be in charge of all code level testing. Why? We should try to keep our automators safe of any biases or influences the code could put in their brains. The minute they start analyzing code and testing against it, they start testing how well the code was implemented. But the code is the interpretation the programmer had about the feature/requirement when he wrote it, and hence we are losing something critical about the testers, they test to make sure users will gain value from the product.

Let me elaborate more the kind of bugs you could miss if you focus your testing in code analysis. Below some examples of why we would miss bugs even with high code coverage:

- **Missing code:** bugs related to negative tests, concurrency or fault tolerance could be missed. Some logic could be required to handle specific inputs to the program. For instance, for handling cases like divisions by 0, we would need specific code for error handling, but if we fail to include this logic, we will hit this bug when introducing 0 to the system. We would get 100% coverage if we introduce two valid inputs (e.g. 2 and 3), but regardless of it we would have a bug due missing code.

- **Requirements not met:** exercising the code with our tests doesn't guarantee that the code is doing what the user wanted. Requirements could have changed or could have been misunderstood and these bugs won't be found by looking at code coverage metrics, they need judgment from someone.

- **Parafunctional bugs:** Bugs related to performance (both responsiveness and use of resources), usability, etc. could be missed even with high coverage metrics. If the program gives you a right answer for $2 + 2$, it could stop being acceptable if it takes 1 day to give you that result.

If testers start looking at the code with developers, even to try to teach them how they can test it, they will become white box testers, and will be biased by the knowledge they gain by looking at the code.

The middle tier is the layer that includes most of the automated business-facing tests at the API level. I like these tests since they are Behavioral Driven. It is not purely BDD as the tests are done after the development is done, because of that many have chosen to call it BDT (behavioral driven testing). Here's the tricky part, who should be responsible for each? Some teams have their automation engineers doing BDDs at the API level, and I think this is the right thing to do. They look at the API as a black box, they don't know how they were implemented, they just know they have to build test scenarios and use those APIs to get the value they need. This level of automation is still black box, so testers (automators) are safe from biases. There are some that argue that the development team should be in charge of the steps definitions (low level programming) and that the Automation team should be responsible of defining the scenarios. I believe that the automation team is capable of writing the step definitions and actually our black box manual testers could help defining the scenarios. It all depends on the skill set you have in your team, but to me developers should focus on product development and unit testing. Communication

across teams is critical for sure, developers need to understand what changes will make things break for the automation team.

API testing is rapidly gaining popularity and adoption across many of our clients, and we are advocates of its adoption too. We've seen how API level testing helps reduce the level of GUI level automation or even manual testing.

Up in the pyramid we see GUI level automation. These tests are the ones done through the GUI, the ones that actually operate and manipulate the presentation layer. GUI level testing is important but they will represent the lowest ROI since they are usually more expensive to build. These tests will serve you well for installation, compatibility based testing, performance and a BVT.

And finally we still see manual session based testing. I'm not sure that not even allowing it to have some space in the pyramid is the right thing to do, since I still consider this is a must have role in the team. You should not insist that all your testers are programmers. I've seen it before where people want to move towards having only automation engineers and they start forcing automation training to all their manual testers and I don't think this is the right thing to do. If you do that, you may have successful transformations, but you may also end up losing great testers that are not interested in programming, some of these could be your most experienced, subject matter experts that served you well before.

Black box manual testers are sometimes the best at sympathizing with end users and identifying threats to the value they are trying to get from the product; they are good at identifying test scenarios and risks from a variety of quality criteria in a context-driven way. These testers usually find ways to test the product that reveal important bugs that must be fixed, but that are not necessarily worth automating (you should not aim to 100% automation, but that's another discussion). These scenarios can come from a variety of heuristics (maybe even UX related), exploratory tests against the product, explicit or implicit requirements (they are really good at identifying things that were not said or written but that need to be included in the product), customer support data (when validating customer reported bugs), and even by studying competitors. These testers like to observe, they interact with the product in specific ways to collect specific observations that they will evaluate and judge later so they can give you quality related information that is relevant and could put the success of the product at risk. They are sometimes the difference between building things right, and building the right thing.

Are you willing to lose them because they are not motivated by writing code?

The best thing that could happen to you is that your experienced black box testers happen to also like writing automation code. If that happens, cheers! Don't let this person go, but carefully manage the balance between exploration and automation, and keep this person away from code biases.

Bottom line, I agree with the pyramid approach and that successful software projects need heavy investment in unit testing and automation at the different tiers, but manual testers will probably never disappear or stop being important. A good development process that promotes communication in all stages (PO, stakeholders, and engineer team) may help reduce the risk of less manual testing, but make sure your process somehow fills the gaps you may create by removing an experienced manual tester.

Examples of the gap could be:

- Product and Product domain knowledge that help identify implicit or missing requirements.
- UX: Learnability, operability, accessibility, error handling, consistency, responsiveness (time).
- Human judgment about consistency within product, with comparable products, with image, with history.
- Risks identified while exploring and exercising the UI and,
- Test refactoring as the tester empirically gains knowledge from the product and changing requirements.

These last two are important. If you force your testing team to design tests and automate them at early stages of development, you are not giving them sufficient time to really think about the risks or coming up with important test *scenarios (some scenarios are probably not even worth automating). Will they come back to enhance their coverage once they learn more about the features? Or will they only have time to go back and invest time in maintaining what they originally automated based on changes done in either requirements or code that affected their automation framework?. Food for thought.

*More on Scenarios (extracted from "An introduction to Scenario Testing" Cem Kaner, June 2003):

- Test Scenarios are based on a story about how the program is used, including information about the motivations of the people involved.
- The story is motivating. A stakeholder with influence would push to fix a program that failed this test. (Anyone affected by a program is a stakeholder. A person who can influence development decisions is a stakeholder with influence.)
- The story is credible. It not only could happen in the real world; stakeholders would believe that something like it probably will happen.
- The story involves a complex use of the program or a complex environment or a complex set of data.
- The test results are easy to evaluate. This is valuable for all tests, but is especially important for scenarios because they are complex.



Juan has been involved in software testing since 2004, and likes it more every year. He continues to be both a teacher and a student as much as he can. Juan started in Jalasoft when they were 25 engineers and he has seen it growing to more than 600 now.

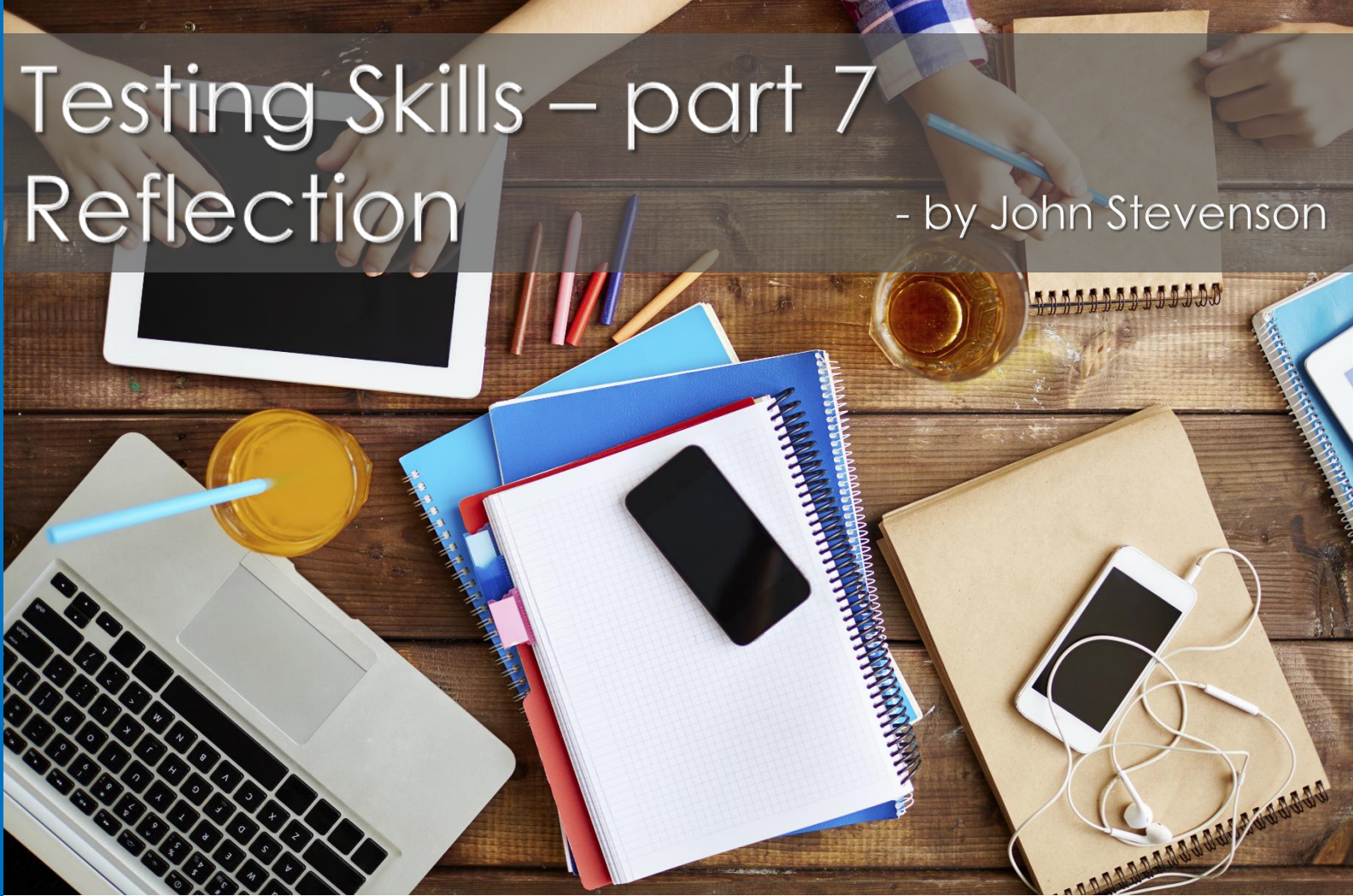
Juan has participated in coaching testers, leads and managers as well as defining QE and Automation career roadmaps. He was able to participate in dozens of projects from many different companies, which allowed him to see many different ways of doing things.

Juan is currently a QA Staffing Director at Jalasoft. www.jalasoft.com

Testing Skills – part 7

Reflection

- by John Stevenson



What type of learning do you need to engage in today?

Do you need to learn facts?

Or

Do you need to analyse and critically think about what you need to learn or have learned? Neither of these ways of learning are wrong and each has its place in your learning journey. Knowing which to use and when is a great tool for a tester to have.

Learning facts is defined as a 'surface approach' style of learning and writing and thinking critically about what you have learned can be defined as a 'deep approach'.

Reflection is important in software testing and you need to be able to communicate what you have found and uncovered when testing. If you have no opportunities, in your organization, to be able to communicate with others then you may find you have a lot on tacit knowledge but little in the way of explicit. It is vital to set aside time for members of the team to discuss what new information they have uncovered or learned when testing. This allows

them to use reflective reasoning to turn tacit into explicit and gives the person providing the information a sense of purpose and achievement in what they have learned.

To improve your own self-reflection produce a reflective action plan. By doing this you are committing to yourself what you intend to do and engage critical thinking skills in doing so. Make sure your plan has goals and deadlines that you can commit to. Make this deadlines and goals public it encourages you to keep to them. Using a [personal Kanban board](#) can help you achieve this.

"First, the deep and surface approaches are not personality traits or fixed learning styles. Students adopt an approach which is related to their perceptions of the task."

[Marton and Säljö - Student Approaches to Learning - University of Oxford](#)

When you need to think deeply about what you have been studying then writing down you own thoughts and understanding is a good way for you to be able to see what you have remembered. If you only need to learn information or facts then other models would be more suited. Writing down your thoughts on what you have been attempting to learn enables you to better remember or apply what has been studied. Reflection is about doing your own internal self-assessment of what you have been learning. This assessment enables you make what you know internally, tacit knowledge, and attempting to make it clear and detailed explicit knowledge. To do this you need opportunities to talk to others verbally or by writing down what you have studied to invoke critical thinking.

"In general, writing appears to be suitable for tasks where the aim is fostering understanding, changing students' conceptions and developing their thinking skills, but less suitable if the goal is the simple accumulation of factual information"

[Conceptualizing and Measuring Knowledge Change due to writing](#)

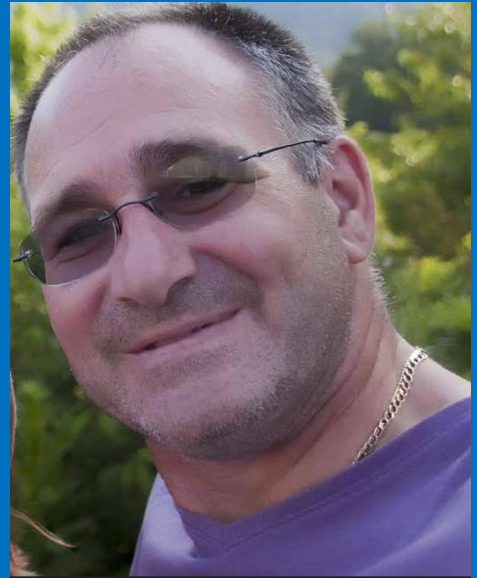
[Schumacher & Gradwohl Nash \(1991\)](#)

One way in which can improve your learning is to use reflection:

"Reflection is an active process whereby the professional can gain an understanding of how historical, social, cultural and personal experiences have contributed to professional knowledge and practice "(Wilkinson, 1996).

You learn from your experiences and to make this happen you need to engage in reflection. If you think about what you are doing or what you have experienced you are engaging in self-reflection. This approach helps you to turn learning into knowledge and it is vital in improving your own knowledge and skills.

Reflection is really about looking back at a situation, thinking about it, critically and learning from the experience, then using that new knowledge to help in future similar situations. This in many ways is similar to the approach you will taking when engaging in testing activities.



John is tester, [blogger](#), [tweeter](#) and [author](#) who has a passion for the software testing profession. He is keen to see what can be of benefit to software testing from outside the traditional channels and likes to explore different domains and see if there is anything that can be of value to testing. At the same time he likes to understand the connections between other crafts such as anthropology, ethnographic research, design thinking and cognitive science and software testing. He is currently writing a book on this called "[The Psychology of software Testing](#)".

John has presented workshops and presentations at various events such as Agile Alliance, CAST, Testbash and Let's Test.

"The process of reflective writing leads to more than just a gain in your knowledge; it should also challenge the concepts and theories by which you make sense of knowledge. When you reflect on a situation, you do not simply see more, you see differently. This different way of viewing a situation is reflected in statements about a commitment to action. Action is the final stage of reflection. "

[Learning through reflection \(Word doc\)?](#)

[Back To Index](#)

Agile Testing Days 2016 Special episodes of Techno-talks with Lalit



Michael "The Wanz" Wansley special episode - Techno Talks wi



Techno Talks with Lalit
Episode 4



Talking about Coverage with
Matt Heusser - TTwLalit



Dimensions of Testability with
Maria Kedemo - TTwLalit



AGILE TESTING DAYS

EUROPE'S GREATEST AGILE
SOFTWARE TESTING EVENT!

November 13-17, 2017, Potsdam, Germany



Australia's Premier Software Quality Conference

10th - 12th
May
2017

Melbourne
Australia

Full day
world class
workshops

Mike
Lyles

2 days of
software
quality
talks

International
industry
experts

Locally Organised

Tickets
from
\$150
per day

Robert
Sabourin

Smita
Mishra

Find out more at
www.qualitysoftware.com.au

Love to
Write?



Write For Us

Your ideas, your voice. Now it's your chance to be heard!

Send your articles to editor@teatimewithtesters.com

A photograph of a classroom scene where several students are raising their hands. The students are seen from behind, wearing colorful shirts (light blue, red, orange, green). They are facing a dark chalkboard. The text 'In the school of Testing' is overlaid in large white font. The entire image is framed by a thick black border.

In the school of Testing

for your better learning & sharing experience

Improving Test Processes by improving Test Data

- by Baris Saricalioglu



Since software testing is mostly a data-driven activity, and testers spend at least half of their efforts in managing the test data (creating, obtaining, using, storing, refreshing, masking, subsetting, and tracking), I want to write about this subject and try to give you some practical ideas and information.

Let me start by telling you a little bit about the “big data” phenomenon. Several investigations indicate that more than half of global companies are storing more than 100 terabytes (TB) of data. And these data are expected to double in the next two years’ time. By 2020, it is assumed that there will be 20 billion connected devices. Apart from these, companies still find themselves immature about handling their data (which is quite correct), and almost all of them are willing to invest in this area for a better governance.

While companies are making the use of these information oceans and derive profits from the data they store, at the same time they suffer from it. It is obvious that no company can cope with data growth by just increasing their hardware capacity. Companies need to find smart solutions for this inevitable growth.



When we narrow the subject to testing, we observe that IT organizations are deeply focusing on the collection and organization of data for their testing processes. The ability to control this process and use test data has become the key competitive advantage for these organizations because benefits of such mechanisms will outweigh their disadvantages. Ultimately, test data management plays a vital role in any software development project. Unstructured processes may lead organizations to:

- Do inadequate testing (poor quality of product)
- Be unresponsive (increased time to market)
- Perform redundant operations and rework (increased costs)
- Be noncompliant with regulatory norms, especially on data confidentiality and usage

We all know that testing is a very critical part of good software development; nevertheless, test data management gets only minimal attention from us. Why is that? I really do not know.

Maybe it is because we are more focused on execution rather than preparation, or we are more focused on the result of the game rather than how we are playing. Whatever the root cause is, we need to accept that many test failures are caused by inconsistencies in the test data. Therefore, we need to be sure that we have constructed an efficient test data management process.



If we do so, then we can be quite certain that test metrics are not biased and we do not cause any interruptions and time loss through our test execution process. On the following page, I list the most essential activities and processes to achieve a complete test data management process. If you follow them in a structured manner, then you can talk about test efficiency, cost reduction, and acceleration. Here they are:

- Initiate a demand tracking process for managing the test data demands and their status
- Including creation of the workflow and utilization of a activity tracking tool
- Include test data analysis activities in the test planning phase
- Specify all the necessary data parameters, like depth (amount of data), breadth (variation of data), scope (relevancy of test data to the test objectives), sensitivity, and architecture (physical structure of the test data)
- Set and frequently measure your test data objectives
- Reliability, accessibility, completeness, consistency, integrity, timeliness, security, and so on
- Include test data preparation activities in the project plan and test development phase
- Estimate efforts for test data analysis and preparation
- Follow a step-by-step approach (below you can see the high-level picture)
- Extracting (from the source database; you can do sub-setting if necessary)
- Masking (for desensitizing the test data and ensuring that there exists no confidential customer information that is against any legislation or legal enforcement)
- Loading (into the target database)

When we go even deeper, we shall observe that every different testing activity (e.g., test type or test level) requires different test data. The following chart hopefully will help you in determining what volume and variation of data you need while you are executing different levels and/or types of testing.

Type of Testing	Test Data Requirements	
	Volume (Depth)	Variation (Breadth)
Unit Testing	Small	Simple
Integration Testing	Small	Simple
Sanity/QA Check	Small-Medium	Simple
Exploratory Testing	Small-Medium	Wide
Smoke Testing	Small	Wide
System Testing	Medium	Simple
System Integration Testing	High	Wide
Regression Testing	Variable	Variable
User Acceptance	Medium-High	Production-like

To clarify your test data requirements, there are questions you

- What kind of data is needed?
- How much data is needed?
- When is the data needed?
- Who will use the data?
- Where will it be extracted, and where will it be loaded?
- Does it have any dependencies?
- Is it sensitive, and how will the data be secured?
- How will the governance be done?
- How will the data be refreshed?

Once you have gathered all the answers and you are satisfied, then you can be sure that you are on the right track.

No matter which approach you choose to handle the challenges of the important subject of test data management, the basic requirements for you to be successful are a combination of good test cases and test data along with the proper usage of tools to help you automate extraction, transformation, and governance of the data being used.

If you want to see results, you need to play well in the game. You either will have excuses or results; the choice is yours...



Baris Sarialioglu has over 15 years of experience as an information systems professional. He is highly experienced in software development life cycle, project management, Agile development, quality assurance, and software testing. Baris also has diverse experience spanning several industries, including telecommunications, defense, banking and finance, semiconductor manufacturing, and aviation. Based on this broad experience, he has been involved in several challenging projects and had the chance to work in several different countries. He has written articles and papers on software development methodologies, quality assurance, and software testing, and he regularly attends international and national conferences as a speaker, panelist, moderator, and contributor.

In 2013, he published his first book, *Software Testing Tips; Experiences & Realities*, in which he shared his experiences and vision about various software testing issues. Later-wards in 2015, he published his second book, *Mobile Testing Tips*, where he addressed many important areas as; Mobile Testing Techniques, Mobile Testing Tools, Mobile Test Automation and Mobile Performance Testing.

Currently, he is one of the managing partners of Keytorc Software Testing Services where he holds the responsibilities of delivering test consultancy, outsourced test management, and software testing training.



Outdated Testing Concepts #2

- by Viktor Slavchev

The first stereotype that was branded in my brain, in the first months I started working as a tester was: **"QA is the person responsible for product's quality"** and **"Your job is to prevent bugs from being released in production"** or **"Your job is to test the product so it meets the functional requirements / acceptance criteria"** and so on.

I can now recall that I wasn't really happy with the job title **"tester"**, to me it used to sound like someone who's about to get shot out of a cannon or perform crash tests on a car or something. In other words **"tester"** seemed to me like a bit of an insult, while Quality assurance ... wow that was shiny, at least to me, at that not too distant point in my past, it was rock star title to me, I was **responsible for the quality** of the product and I was told, **I am the man to give my final verdict** on shall it live or we will have to fix bugs until it's done. Seemed like a pretty **bad ass** job to me ... and it was **all lies**.
How are we responsible for quality?

Now, enough rainbows and unicorns so far, we are back in the present and we put on our "investigation hat" and analyze our part in product's quality, and if it is really the principal role that was described to us, or just another outdated concept.

Let's start at the pretty basic level:

- **If the developers are writing crappy code, no unit tests, can we impact on that aspect of quality?**

No, of course we can discuss with them, brag, ask, threaten or whatever negotiation tactics we might use in order to **make** them do it, but we can't actually force them to do it, therefore we are not responsible for that part of the quality, of the software product.

- **If the requirements for the new features in our product are vague and ambiguous, impossible for us or the devs to decipher, do we have the power to change that?**
In fact we can, but it doesn't guarantee anything. Why is that? We can protest against bad written requirements, we can ask for revisions and meetings in order to "sort things out" and make them clear and understandable for the whole team, but let's face the reality, we are in the **21st century**, we are mostly being involved in Agile and Scrum projects, because development teams **like**

changes and they like them **in big portions and at really high-speed**. These poor requirements are probably not going to be the same only in a week, and who says they are the only source of knowledge for the product? What about the emails, or instant messages, other documentation like design blueprints, how about meetings, phone calls, conference calls, hangouts, 1 on 1 meetings or managers who tend to sit next to a developer and tell him "how it is supposed to work"... Do you still think you have reach on all that? I strongly doubt it.

- **If the management takes bad decisions for the vision of the product?**
No, after all we are there to help the product to succeed, but it's not our job to manage it. Of course, we could be involved in management, but there's certain management decisions that could not be argued. And this is the **end of the belief that we have the right of "veto"** over the release. And me personally, I am happy with that, I wouldn't like to bear the responsibility for that decision on my own. **This isn't one man decision for a good reason** – everyone **makes mistakes** and we don't want to turn anyone, not just the tester, into a **sacrificial lamb**. It's easier to have someone to blame and many companies might still do this, but it's not the way to progress.
- **Can we assure the quality of testing in general and/or the quality of our own job?**
This is a really tricky question, but I would say no. Do we really have a way to track the work progress and quality of the other testers, can we guarantee that the job they perform, even if documented, follows **our understanding** of high quality testing? And what's even more critical – **can we guarantee the quality of our own job?** I personally wouldn't, because it will be easy to **delude yourself**, and after all we as testers **must always be suspicious** and even our own skills should be tested and analyzed, through our own senses as well as from the feedback that other members of the development team. Only this way, we can be sure, that the job we perform adds value to the project, not only to our expertise, or as Keith Klain mentioned in a talk I linked [in this post](#), this is a way to make sure **"we are performing our testing holistically"**.

And the list goes on and on... So, as we see, the responsibility about product quality **isn't a single person responsibility, it's a team responsibility**. Not in the terms of "shared responsibility is no one's responsibility", but in the terms of, **everyone adds value with his own unique skills to the product and bears the responsibility for the consequences of his mistakes**.

So, from the picture that I drew so far it seems a bit worrying what we can do, isn't it? Don't worry ...

"There has been an awakening..."

We are no longer happy being called QA and we realize that our real profession is **testing**. **And by testing I don't mean just "play with the product", as we often get thought to believe by non-tester individuals, but the process of analytical, conscious experiment with the product in order to explore it, evaluate it (J. Bach) and provide expert opinion what are the possibilities for it to succeed or the risks that can lead it to failure.**

So, how could we promote our job, from all what I said, doesn't it seems that we all have no chance to change anything? No. Here's a good list of actions we can perform to promote the **real value** of the testing profession:

- We have the most wonderful, interesting profession in the software business – the testing part, we can experiment with the product in order to expose its weak and strong sides.
- We can construct our testing as **complete scientific experiment** and develop not only our testing, but as well our scientific skills and the way we learn.
- We have the opportunity to put our hands on the product first, after the development team is done with it.

- We have the opportunity to mediate between the marketing, business, management and the technical teams within a software project and in this way learn way more information than **any of these departments' representatives alone**.
- We can literally break stuff and no one is angry with us, sometimes we even get congratulated for breaking it. This is a joke of course, we all know "we didn't break the software, it was already broken when it came" ([Michael Bolton](#)).

The list goes on and on. As you see, speaking of software testing as simply "someone being responsible for the products quality" is just naive and barely touches the surface of what software testing profession really is.

And last, but not least, I believe that the main way in which we can drive our profession to progress is:

I believe James Bach once mentioned these...

1. Learn the testing vocabulary and terminology.
2. Learn the testing methodology and techniques e.g. learn how to be experts in our craft and how to provide high quality service as testers.
3. Learn to explain our craft in a manner to promote our profession.

And I believe the last one is **really important**, we need to make ourselves **heard and seen**, we need to learn to give a **compelling story** on what testing is and what is its true value.

That's for this time, I hope it was interesting and useful, stay around for the next part. I will be happy to see your feedback, as always. Good luck



Viktor Slavchev - Senior QA @ siteground.com, from Sofia, Bulgaria.

Currently involved in the area of web hosting, Viktor has previous experience and interests in the area of mobile and web testing, non-functional testing and in the telco area.

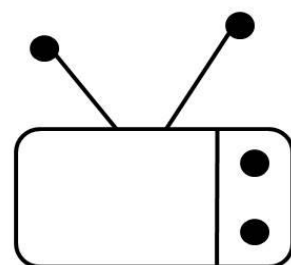
Viktor is a passionate tester, who doesn't believe in "golden rules" about testing and thinks everything has to be put to a test. He likes to explore new trends in testing and learn from them. Strong believer that testing isn't simply a technical, nor soft skill, but can benefit from a lot of humanitarian disciplines like psychology, philosophy (epistemology in particular), sociology, anthropology, history and others.

Viktor is also passionate in teaching testing and helping new testers in mastering the craft, he's currently giving lectures in an local IT academy, called Pragmatic, on various testing topics like Exploratory testing, Mobile testing and Non-functional testing. In his spare time, he's a passionate MMORPG gamer, listens to loud rock music and enjoys reading interesting books. You can read more from Viktor at his blog - [mrslavchev.com](#) or follow him on twitter @Mr_Slavchev



[Checkout all Episodes](#)

TV for Testers



Your one stop shop for all software testing videos



Sharing is caring! Don't be selfish 😊

[Share](#) this issue with your friends and colleagues!



Happiness is....

Taking a break and reading about **testing!!!**



Like our FACEBOOK page for more of such happiness

<https://www.facebook.com/TtimewidTesters>

T ' Talks



T. Ashok exclusively on software testing

Practice (not process) makes it perfect, rapidly.

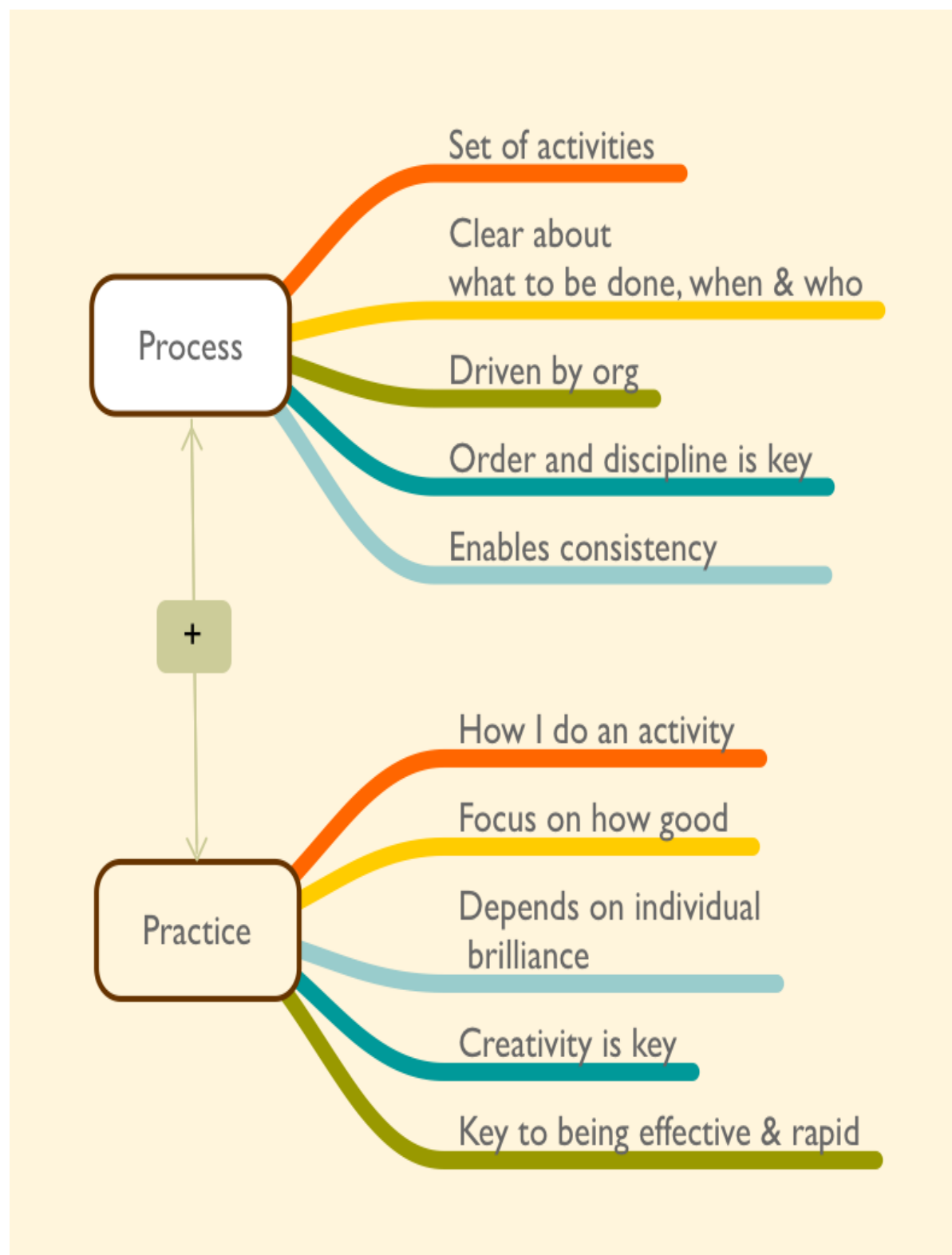
Organisations believe in processes as a means to get things done well. However in recent times, the emphasis has shifted to the individual. It is about 'how to enable individuals' to deliver their full potential. How can we equip individuals to solve problems smarter and rapidly?

Traditionally it has been to formulate and setup processes. Aided by 'templates & tools' it was seen as a predictable way to install order and enable streamlined work. What is a process? It is the set of activities we do, normally agreed upon by stakeholders in an organisation. This view is especially useful when problem solving has matured resulting in a set of standard activities and therefore all we need to do is to ensure good deployment. Is that good enough to solve all problems? No, especially when problem is morphing rapidly and solution to be constantly tweaked.

So in the light of continual changes that require rapid adjustments, the shift has been to make processes light so that they are nimble and responsive. Process works great in enabling overall discipline but the "individual habits" of good practice are vital to being effective and rapid.

Practice on the other hand is how we do these activities, largely driven by individual brilliance. It is about how we breakdown problems, ideate solutions and rapidly implement them. It is about techniques, principles, guidelines (heuristics) that one uses to solve problems.

Process is like the 'route map' whilst Practice is like the 'driver skills' to get to a destination quickly and safely.



Equipping an individual with good method(s) that when practiced continually enables one to deliver great work, the outcome of 'practice'. Mere processes don't cut it, however when good individual practices become common place, they get converted to "Best/Good practice" becoming part of the process.

Good practice is the outcome of intelligent thinking whilst process industrialises a practice enabling it to be deployed mindlessly.

Hers is to wishing to brilliant practices in 2017. Have a great year my friends.

[Back To Index](#) 



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



C Complexity is also a Bug

Last month I attended the DevOps Days Tel Aviv conference, I took some notes about the interesting stuff people said – like many people do during a conference – and then I put the notebook away – like most people usually do after a conference...

But yesterday I was going over my notebook, I guess I wanted to check what I still had to do from my endless to-do-lists, and I stumbled onto my notes from the conference. Among all the scribbles and drawings there was a sentence that caught my eye, something I had written and then forgotten, but that when I saw it again managed to grab my attention.

What was written in this sentence was:

Complexity is also a Bug.

Complexity is increasing and will continue to increase. Below the sentence I wrote the notes that the speaker had said in his presentation.

He was a Founder/Programmer/Geek/CEO from The Valley, and he explained how from his point of view complexity in software and applications was increasing, and would continue to increase with time.

The point he was driving to was that, as development companies we have to make choices about who handles this complexity. He looked at this from a DevOps perspective and so he said that there were 3 choices for who could “take on” this complexity: Development, IT or the End Users.

To put it simply, applications need to do more: talk to more applications in order to do stuff, do this stuff faster, and do it in more diverse environments. We are not really doing less of anything (unless you count less sleeping and relaxing).

Being this the case, someone needs to “pay the penalty” for doing more, and this penalty could be handled by any of the 3 teams I mentioned above.

It could be handled as part of the development process, by creating more complex algorithms and smarter code to handle all the complexity.

Parts of it could be passed on to the IT, who need to deploy and configure the system so that it could handle some of this added complexity.

And finally, all the stuff not handled by any of previous two teams would end up being passed to the End User, who would need to work with more complicated installations and configurations, and applications that were less friendly.

In the end, what he was explaining is that complexity is a [Zero-Sum game](#), and as part of our development projects we need to decide who will be handling it.

Looking at complexity as a bug in the product

Complexity is not a feature, it is also not an objective attribute like load level or application response time, but it is not less important than any of the other attributes in our products, such as a nice GUI or a good [UX](#). Actually, complexity is usually handled via a combination of features and UX solutions. Having said that, we need to handle excessive complexity as a bug in the products we are testing, but this may sometimes be trickier than you think.

- **First of all, complexity depends on the user.**

Try to think about 2 different users: for example your dad vs. yourself. My father is a great surgeon, but he is pretty bad with iPhone apps. This means that something that is trivial to me may be impossible for him to figure out.

On the other hand, if you compare me with some of the developers we have in PractiTest, then I become the guy who cannot get most of our configuration done straight. In this second case, something that to them may be trivial for me is utterly complex.

- **Second of all, complexity is a trade-off.**

I think we have all been in those projects where the Product Owner or Product Manager or even the End User comes and says: "Guys, we need to release this! It's time to make compromises..." Compromises are just another term being used for bugs and missing features. These are the cases when the project is already delayed and we need to deliver something out, even if it is not perfect. And in this cases, one of the first things to be "compromised" is complexity. This is when we agree that we can release a document on how to configure the system manually instead of having the intelligent self-configuration, or that the installation will ask many questions that could have been taken from the system directly.

These are not critical things, but depending on who the user is this may be a blocker for him/her to install the system. Or in the best of cases only a nuance they would have preferred do not deal with.

- **Finally, it is very hard to handle complexity after the feature was developed.**

Imagine you order a car to be custom made for you. You talked about the color, the shape, the interiors, wheels, mirrors, etc.

Then, when they are showing you the finished car you realize that (1) you wanted this car to be driven in the UK where they ride on the Right Side of the road, also that (2) you wanted to save money by having it work with a Diesel engine, and finally that (3) you wanted an automatic car, and the one they are showing you works with a stick...

Oops!

Just as making these changes once the car is done will be very costly and it will definitely delay the time you get to drive your car... So is the case with trying to reduce the complexity of the system once it has been done and it is close to being released. It is true that by [working AGILE](#) and with Iterations these issues should be reduced, but they will not be eliminated. And many times the issues won't be discovered until the product actually reaches the end user, and so solving the complexity issues or adding new functionality to handle them becomes even more costly and may end up having additional marketing repercussions and costs.

Complexity is an issue that needs to be taken seriously into account



As you may understand by now, complexity is not only here to stay but it will become more of an issue as our products need to do more and communicate with additional products and solutions in order to fulfill their objectives. Complexity is also something that, if we don't handle it as part of the product design and development, it will be passed to our end-users to handle – or they may choose not work with our solutions because they don't want to or can't handle the complexity we are relegating to them.

And finally, complexity is easier, faster and cheaper to handle early in the process than later on.

But most importantly, given that your users will see complexity as a flaw in your product, it is better that you start considering it as a bug that needs to be reported and eventually handled by your team.



Joel Montvelisky is a tester and test manager with over 15 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](#)



#ATAGTR2017



CONFERENCE

16-17, MARCH 2017

POST CONFERENCE WORKSHOP
18-20, MARCH 2017

Speaker Benefits



Exhibit



Interview



Presentation



Free Entry



Lunch



Certificates



Trophy

GTR is one of the largest, best and most fun filled global testing conference.

Be a part of this conference by submitting **Papers** or registering now for early bird benefits

🏠 gtr.agiletestingalliance.org/papers.html

☎ +91 84510 72226

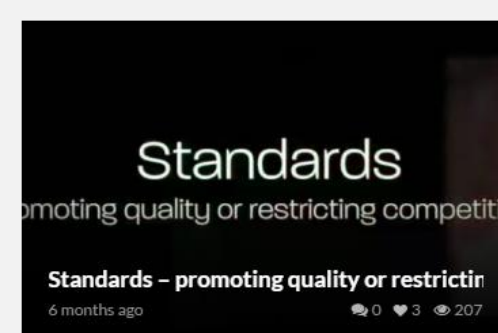
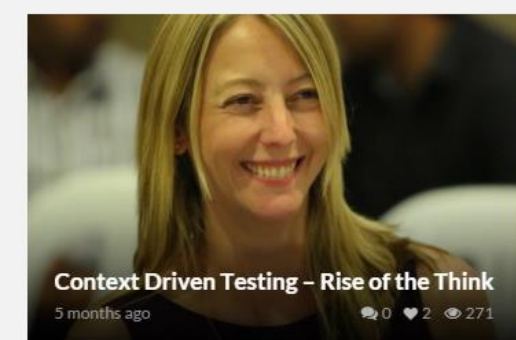
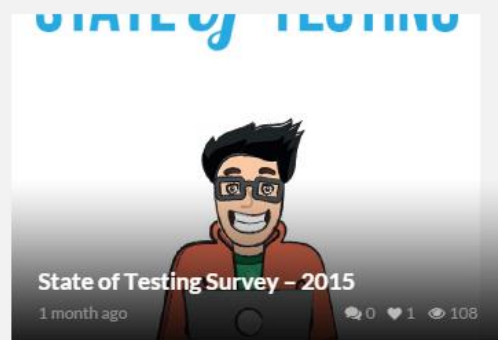
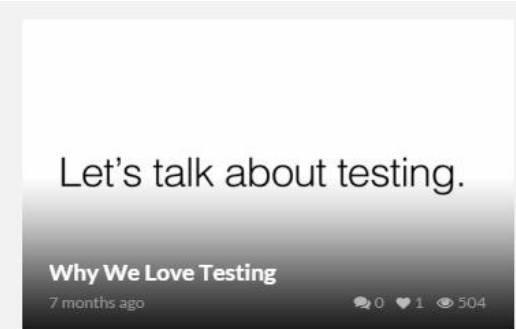
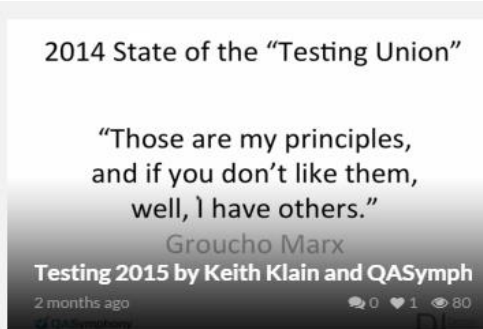
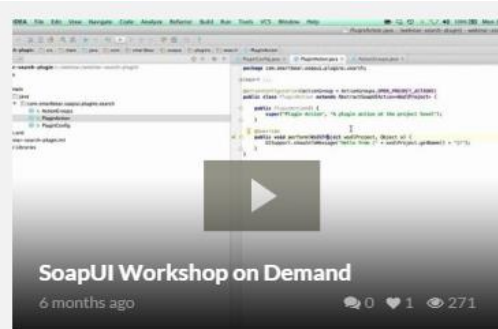
✉ atagtr2017@globaltestingretreat.org

gtr.agiletestingalliance.org


Got tired of reading? No problem! Start watching awesome testing videos...

TV for Testers

Your one stop shop for all software testing videos



WWW.TVFORTESTERS.COM



www.talesoftesting.com

What does it take to produce monthly issues of a most read testing magazine?
What makes those interviews and articles a special choice of our editor?
Some stories are not often talked about...otherwise....! Visit to find out about
everything that makes you curious about **Tea-time with Testers!**

Advertise with us

Connect with the audience that MATTER!



Adverts help mostly when they are noticed by **decision makers** in the industry.

Along with thousands of awesome testers, Tea-time with Testers is read and contributed by Senior Test Managers, Delivery Heads, Programme Managers, Global Heads, CEOs, CTOs, Solution Architects and Test Consultants.

Want to know what people holding above positions have to say about us?

Well, hear directly from them.

And the **Good News** is...

Now we have some more awesome offerings at pretty affordable prices.

Contact us at sales@teatimewithtesters.com to know more.





Every Tester

who reads Tea-time with Testers,

**Recommends it to friends and
colleagues .**

What About You ?

in ne>xt issue

articles by -

Jerry Weinberg

T. Ashok

Joel Montvelisky

Viktor Slavchev

...and others

our family

Founder & Editor:

Lalitkumar Bhamare (Pune, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

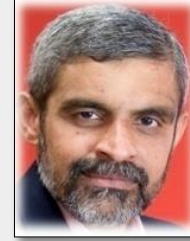
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover page image – Metaphrasi

Core Team:

Dr.Meeta Prakash (Bangalore, India)

Dirk Meißner (Hamburg, Germany)



Dr. Meeta Prakash



Dirk Meißner

Online Collaboration:

Shweta Daiv (Pune, India)



Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)



Kiran Kumar



Chris



Romil

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

To get a **FREE** copy,
[Subscribe](#) to mailing list.

SUBSCRIBE

Join our community on

facebook.

Follow us on - [@TtimewidTesters](#)



www.teatimewithtesters.com

