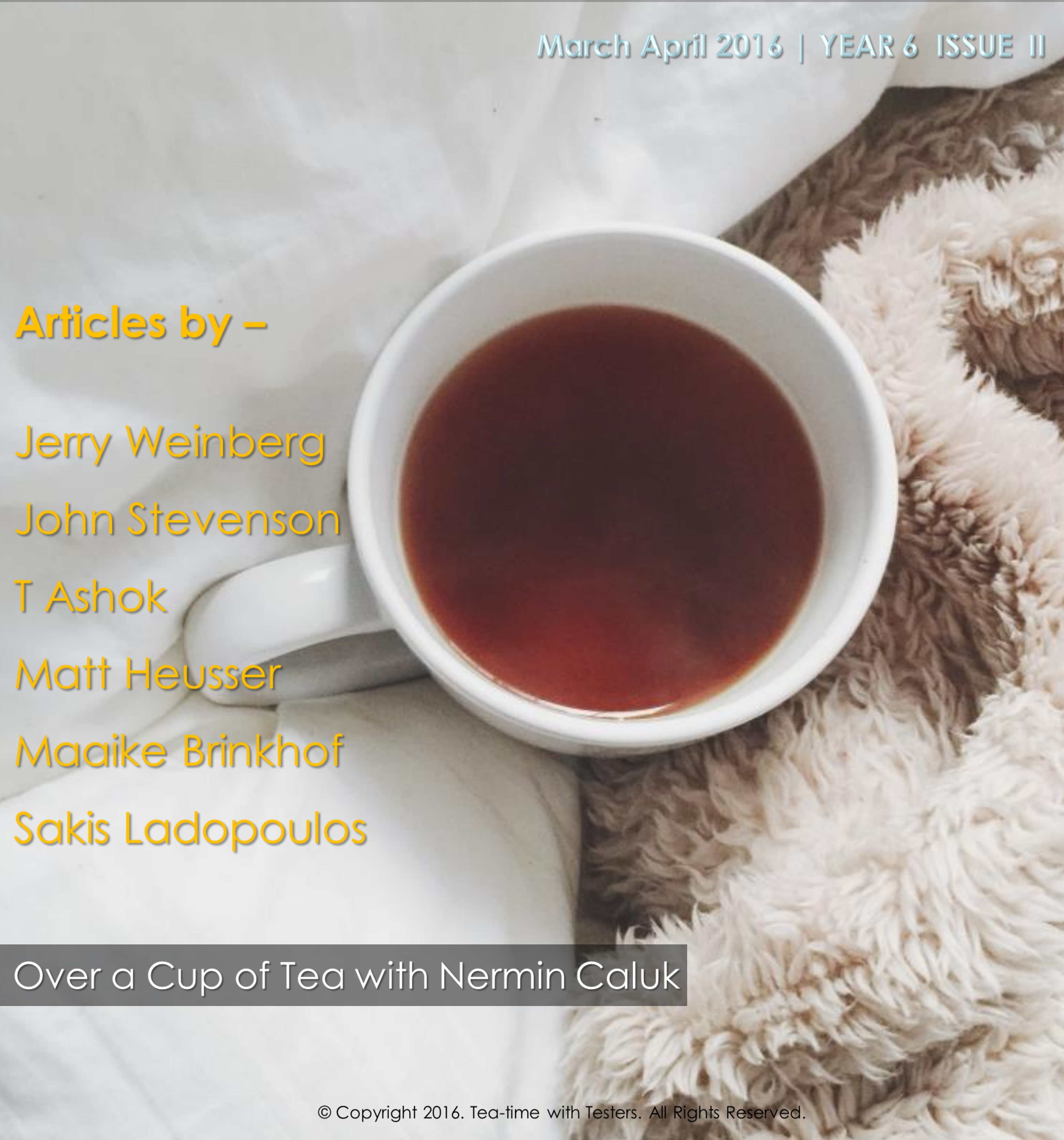
THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

March April 2016 | YEAR 6 ISSUE II



Articles by –

Jerry Weinberg

John Stevenson

T Ashok

Matt Heusser

Maaike Brinkhof

Sakis Ladopoulos

Over a Cup of Tea with Nermin Caluk

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: (+49) 15227220745
Pratik: (+49) 15215673149

Editorial

The Unsung Heroes

In last issue I mentioned about "Unsung Heroes" as a theme for our interviews.

So far we have been publishing interviews of experts and passionate testers who are mostly active in testing community so that their ideas, innovations reach out to more testers. We'll continue to do so without a doubt. However, from now on, in addition to that, you will get to know about testers who are not very visible to community at large but deserve recognition.

The reason for that is very simple. From my connect with testers from within community and through personal interactions, I have come to know some passionate and brilliant testers that are not very visible to outside world (since they don't blog often, are not frequent visitors to conferences, are not on twitter or for any other reason) but they have brilliant ideas, experiences to share, points to raise and contributions to make. These are some testers I have seen making big impact on shaping intellectual testing culture in organizations they work for, inspiring others to take actions and making valuable contribution to testing community by helping implement what experts in our field teach. And that is not easy.

Yes, I strongly support that all such testers should become more active in community, they should write blogs or exchange their knowledge on some platform. But again that's not something we can expect everyone to enjoy doing. It's an individual's choice in the end. Most importantly, not doing all of these does not make the contribution they are doing less important. What matters more to me is that people are willing to contribute and are contributing in ways they are comfortable with.

Through Tea-time with Testers, you will come to know of such unsung heroes once in a while. They have stories to share and also deserve attention. Let's hear them out.

Sincerely Yours,

[illegible] - **Lalitkumar Bhamare**
editor@teatimewithtesters.com
@Lalitbhamare / @TtimewidTesters

QuickLook

LST News

All About Software Tools

What's making News?

The Pentagon's official F-35 has terrifying bug list

Source: Joel Hruska on Extreme Tech

The first contracts to design what became the F-35 were handed out 20 years ago. Lockheed's X-35 won the contract in October, 2001. Fifteen years later, the aircraft is in terrible condition — a fact driven home by the DoD's own official report on the state of the F-35 and the bugs that continue to plague it.

The F-35 undergoing wind testing

The report was released few months ago, but a number of links to the PDF have died; though. It discusses all variants of the F-35, but focuses on the F-35B, the short-takeoff-and-vertical-landing version of the aircraft developed for the US Marines, and adopted by the Royal Navy as well as the RAF.

Check out what all terrifying bugs F-35 had in this interesting news report here.

UNSUNG HEROES

INTERVIEW

Meet Nermin Caluk, Engineering Team Lead at XING AG and also a passionate Context Driven Tester.

Based what I have seen Nermin doing and after discussing testing with him very often, I genuinely feel that we need more testers like him in community and organizations alike.

My discussions with Nermin made me realise his passion for testing and also his deep understanding around implementing Context Driven Testing at work.

I feel that Nermin is perfect example of what we intend to do with "Unsung Heroes" series in our interviews. And I am glad that we are starting it off with him.

Read his interview to know more about his ideas about testing and also to know what makes me say what I said about him…

- Lalitkumar Bhamare

Over A Cup of Tea with Nermin Caluk

It's been pleasure meeting you, Nermin. And thanks for talking with us today. Congratulations on your new role at XING!

Thank you for this opportunity, Lalit.

Well, I'm curious. How do you look at your switch from highly experienced test engineer to an Engineering team lead now?

In my view, proper software testing and development are much closer than it is often portrayed and taking different technical roles within software engineering can help better understand the complexity of software products and processes (and professionals). It resembles switching the player position on the court: everything your previous position taught you is very much useful and shapes you in a unique way.

Do you think role of a tester in our industry is changing? In what ways if yes?

Tester's role should have changed faster in the last 10 years. Testers should focus primarily on activities that contribute software quality [the most, in a given context]. It is that simple. Ideally, the modern tester should not to be perceived as someone who slows down things, but as a catalyst.

How do you think "an engineer with testing mindset" differs from pure-play programmer or traditional tester?

Developers who realize the importance of testing are much more responsible than those who carelessly throw code changes over the fence and wait for someone to test them. Actually, while developing, developers do much more testing than we usually think: every time they run their rails app locally or their android app in the emulator they are testing the latest code changes.

Building something without testing feels wrong: imagine a restaurant cook who never tries his food before serving it to customers. That kind of cook unknowingly makes more mistakes, learns slower and can be dangerous to the business.

It's understood that with the advancements coming in Agile way of working, expectations from testers are changing too. Do you think that it can compromise the value a tester with independent mindset can add to the team otherwise?

There is a difference between an in-team tester and an external auditor. In agile environment, the closer the tester gets to developers, the more exciting the job can get. If a tester acts like "we are all on the same side" in the team, he or she can strongly influence the processes, pre-coding activities (preventing problems), testability, releasing strategy, live systems monitoring etc.

Actually, this very broad field in which a tester can contribute to better quality is what I love about the testers' role at XING and I do not think that closely connected roles prevent testers from doing their job. Would you rather have a screaming independent tester without influence or an in-team tester with strong impact?

What are the traps set for testers working in Agile environment?

Because testing and tester's role are so context sensitive, there is no universal formula on how to be an ideal agile tester. Successful agile tester in one company could seriously fail in a different company if they do not synchronize with the new environment, which does not mean they should take everything for granted of course. I would say the biggest trap, not only in agile environment, is ignoring the context.

And how can they still contribute their best?

Almost all software companies play the game in a very competitive market - exceptions to this may be some protected R&D teams, monopolistic companies or some government services. This means that testers need to be comfortable playing on the thin line between testing and releasing, that is, they are aware of risks and as soon as they are acceptable the testers should encourage releasing software changes to shorten the time to market. Many testers in the past were slowing down the pace of releasing software more than necessary. That is not good for the business. Sure, there are certain environments where the rules are different, but I always encourage testers not to put *testing* in the center of their world, but to let the *release* be in the focal point and then everything else around comes naturally. The event of shipping working software to the user is where it counts, so testers can contribute a lot if they are aware of this.

I believe that you firmly believe in principles of Context Driven School of Testing. What according to you is CDT in Agile/SCRUM context?

CDT and agile actually have some overlaps - just consider the CDT principles #3, #4 and #5 and compare them to Agile Manifesto. Also, both CDT and agile are applicable in more general situations, not only software development: try replacing "project" with "life" in the CDT principles; or try applying agile to non-software projects. Because of that, they are hardly separable.

Your question deserves significantly longer analysis, but from a practical standpoint I think today's testing professionals must understand the essence of both. They can print out the principles and the manifesto, put them next to their bed to read them every morning they wake up, still they need to understand them on a deeper level and apply them.

In testing community at large, there are many tester who work from the offshore or under 'multi-vendor' scenarios. Naturally, they don't get to experience the real feel of working under Agile/SCRUM methodologies. What would be your advice for such testers who are underprivileged but not less competent?

It does not mean they are underprivileged, nor that agile is a privilege (rather it is a necessity in some companies that want to remain profitable). However, in many companies the tester's position is more stressful than developer's because certain business models do not optimize for quality, yet testers are held accountable for it. For example, many agencies must sacrifice a lot in order to meet the tough deadlines.

Some offshore studios cover their liability with heaps of documentation, so when some obvious bug pops up: "well, you did not define that test case". Some contracts provide for shipping poor quality v1.0 software because it is cheap, after which the company charges a hefty sum for years of maintenance. There are contracts where testers are paid per bug found and this is so wrong because if the developers are in the same company there is a huge conflict of interest, and if not then the whole atmosphere is rather hostile and you know consequences of that.

In these and many other examples you can see that certain business models are not nice to testers at all, let alone to enthusiastic agile testers. Applying agile in a product company is often easier than in the above described environments. When not in product companies, testers should try to be in environments where they sit together with developers and help each other, and externally, they should connect with the testing community, e.g. with Tea-time with Testers contributors.

If you are open-minded, agile is easy once you are in an agile company and in the meantime simply use every opportunity to learn and to deepen the technical knowledge.

You have had played an instrumental role in building intellectual testing culture at XING. We are curious to know about your experiences and lessons learned.

There are many great professionals in the company who share similar views of how development and testing should work. I tried to steer and optimize the hiring process to support the growth and get even more enthusiastic testers on board. On the testing side, I tried creating a culture of helping each other (test sessions), sharing (tech talks), adopting modern ways of testing (*lean*er, mind maps instead of long documents, CDT, testing in production) and encouraging testers to see and shape the whole picture from the concept to the live system.

One of the lessons learned is that hiring is one of the most important processes, especially in growing companies, yet it is often neglected and underestimated. People determine the culture of the company, not vision/mission/values, and you will not get too many great people if you do not invest in hiring.

Who according to you is 'experienced tester"? (You had asked me this in my interview and now I'm curious to know about your answer for the same question :-))

Well, I am curious to learn what different people understand by, for example, "senior engineer" because this worldview tells a lot about themselves, too. In my opinion, many characteristics of senior engineers are universal across different engineering fields, from electrical, mechanical, chemical, civil engineering to agricultural and aerospace. Senior is much more than just years of experience. Seniors have seen projects (or companies) fail and even better: helped save them. Their behavior bonds the team and creates trust; they help include and develop the rookies. They show initiative and do not wait the work to land on their desk.

They never stop experimenting and learning. Seniors can "disagree and commit". They understand the business model behind the company and know the competitors. And let's not forget: they are aware of codes of engineering ethics with consideration for the public, clients, employers and our profession.

What would be the expected skills if a tester wants to work in your team?

Professionals who go successfully through the hiring process for my teams have certain things in common: they are open-minded and nice people who easily fit the team, they have healthy motivation (which often correlates with the fact that they stay for years in the company, plus they show more initiative) and they are hardcore at learning. When these characteristics are there, we talk about the tech skills. In that area I love to see people who understand at least basics of software architecture as this is essential when deciding how to approach testing and how to narrow down where the causes of detected problems lay. In my interviews we do at least one testing exercise where I simply want to see the ability to generate relevant test ideas and how their brain works.

Years ago I used to ask those "How would you test…?" questions where you discuss testing a random object, but seeing a person interact with software is even more revealing. Would you hire a juggler without seeing them perform? I have seen people applying for testing positions who act like they are afraid of software.

The tools, they can be learned, but if somebody is bragging about their tool arsenal I would like to see at least some open-source stuff there.

You actively send out reading recommendations to your testing teams regularly. What are the blogs, books or magazines that you personally enjoy reading?

People who have influenced my view on software (testing) include Cem Kaner, James Bach, Gojko Adzic, James Whittaker, Michael Bolton, and Joel Spolsky among others.

I also enjoy watching videos about testing, such as tech talks or conference recordings.

Your message to our tester friends would be…

If you find testing wishy-washy, you are doing something wrong.

And my last question for today, what do you think about 'Tea-time with Testers' magazine? We would love to get your feedback.

I like your core purpose (giving voice to community of software testers) because the community strongly shapes this profession.

You live your mission and I can only imagine the effort required in the background to keep producing new issues. Keep going :)

Understanding Testing

Last week, Joe Colantonio interviewed me for his milestone 100th Test Talk. In case you haven't heard it yet, Joe extracted a few quotes and insights from this Test Talk. I've put some of them here, followed by a fascinating supporting story from one of my listeners.

Secrets

Joe asked me something about secrets of how I managed to do so many thing, and I gave a couple of long-winded answers:

Maybe the secret is, a sort of middle level, is stop looking for secrets and just figure out one little improvement at a time. Get rid of things that are using your time that are not productive and that you don't like to do. Part of it is you have to love what you're doing and if you don't love it, then it's pretty hard to bring yourself back to it.

I guess the secret of being productive if there is a secret, is to adapt to what is as opposed to what should be. Of course another way to describe testing is that it's finding out what actually is as opposed to what's supposed to be. A program is supposed to work in a certain way and the tester finds out it doesn't work in that way. When you report that then somebody does something about it. If you live your life the same way you'll be pretty productive.

One Thing Testers Should Do

Joe asked about what testers should be doing that they may not be doing, and one of my answers was this:

I think that you need to highlight certain things, like I need to just sit down and talk with the developers and ask them what went on, what happened, what was interesting and so on in an informal way. This gives you a lot of clues where you might be having trouble.

History of Testing

On the history of test groups, I had this to say:

We made the first separate testing group that I know of historically (I've never found another) for that Mercury project because we knew astronauts could die if we had errors. We took our best developers and we made them into a group. Its job was to see that astronaut's didn't die. They built test tools and all kinds of procedures and went through all kinds of thinking and so on. The record over half a century shows that they were able to achieve a higher level of perfection (but it wasn't quite perfect) than had ever been achieved before or since.

Automation in Testing

Joe's sponsor, Sauce Labs, specializes in automatic testing, so we had an interesting back and forth about test automation. Among other things, I said,

You can automate partial tasks that are involved in testing and maybe many tests and save yourself a lot of efforts and do things very reliably, and that's great. It doesn't do the whole job.

To which, Joe taught me the expression he uses is not "test automation" but "automation in testing." We were in violent agreement.

How to Improve as a Tester

Joe then asked me how I'd recommend someone could improve themselves as a tester:

The little trick I give people is that when you find yourself saying, "Well that's one thing I don't need to know about," then stop, catch yourself and go and know about that, because it's exactly like the finger pointing that we talked about before, when the developer says, "Well that's a module you don't need to look at," that's the one you look at first. You do the same thing yourself say, "That's a skill I don't need, it has nothing to do with testing," then it does and you better work on it.

How Testing is misunderstood

There's a lot more lumps like these in the podcast, but here's one straightforward example that supports most of the things I had to say. Albert Gareev sent me this story after listening to the podcast:

"We are not NASA,"—said a Director of Development.—"My testers just need to verify the requirements to make sure that our products have quality."

What she wanted to "fix" in the first place, was that "the testing takes too long." There were a "Dev testing" phase, and then "QA testing" phase, and then "internal business acceptance testing" phase, then "client business acceptance testing" phase—sometimes even two. If bugs were caught at any point, the updated version would go through the same whole pipeline of testing. The product was indeed doing what's intended for the customers and for the company, with a very low number of incidents that were successfully taken care of.

During my initial analysis I found out that those phases were highly compartmentalized. Certain bugs had a very long life span, before the Dev team would get to know of them. Certain problems kept re-occurring. Testing was performed as a kind of a clueless Easter eggs hunt; Dev team didn't feel a need to share what code module they updated.

It appeared to me, that the product was in a good shape only thanks to these lengthy testing phases that made chances of "stumbling upon" the errors high enough to catch the majority of bugs.

So I brought my findings back to the Director, and made a few suggestions—around collaboration, feedback loops, and especially about training testers to model risks, and to test for them.

But she didn't buy it. She still thought that "they're doing too much testing."

I was puzzled. Frustrated. Even slightly offended, being a tester devoted to his profession.

And then I took a project manager for a cup of coffee.

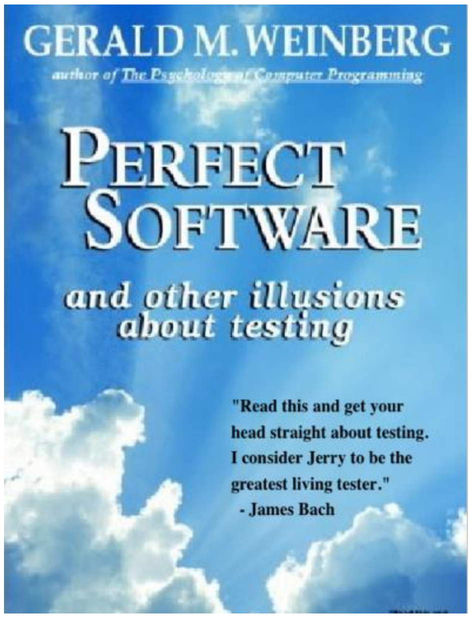
The PM shared that the Director was in her first year being in charge of software development. That previous decade she spent as a director of customer support.

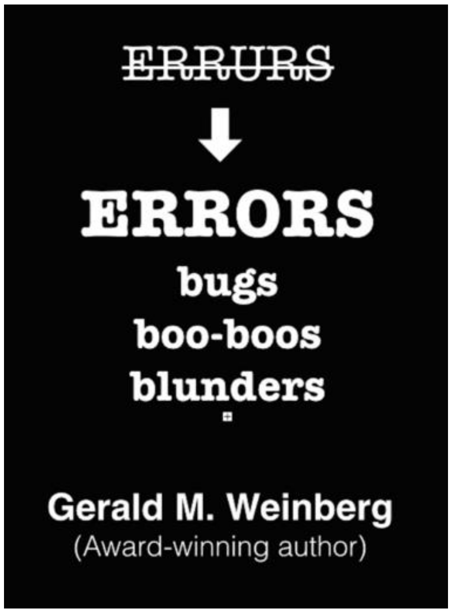
"Huh."—I said.—"All these years she saw the products that already undergone a thorough testing-and-fixing process before her team could try them."

"Exactly."—replied the PM,—"She has no idea that at first it's always messy."

Back To Index

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


GERALD M. WEINBERG
author of The Psychology of Computer Programming
PERFECT SOFTWARE
and other illusions about testing
"Read this and get your head straight about testing. I consider Jerry to be the greatest living tester." - James Bach



People Skills—Soft but Difficult
ARE YOUR LIGHTS ON
What Did You Say? The Art of Giving and Receiving FEEDBACK
Exploring Requirements 1: Quality Before Design
Exploring Requirements 2: First Steps to Design
BECOMING A CHANGE ARTIST
MORE SECRETS OF CONSULTING
Becoming a Technical Leader

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.

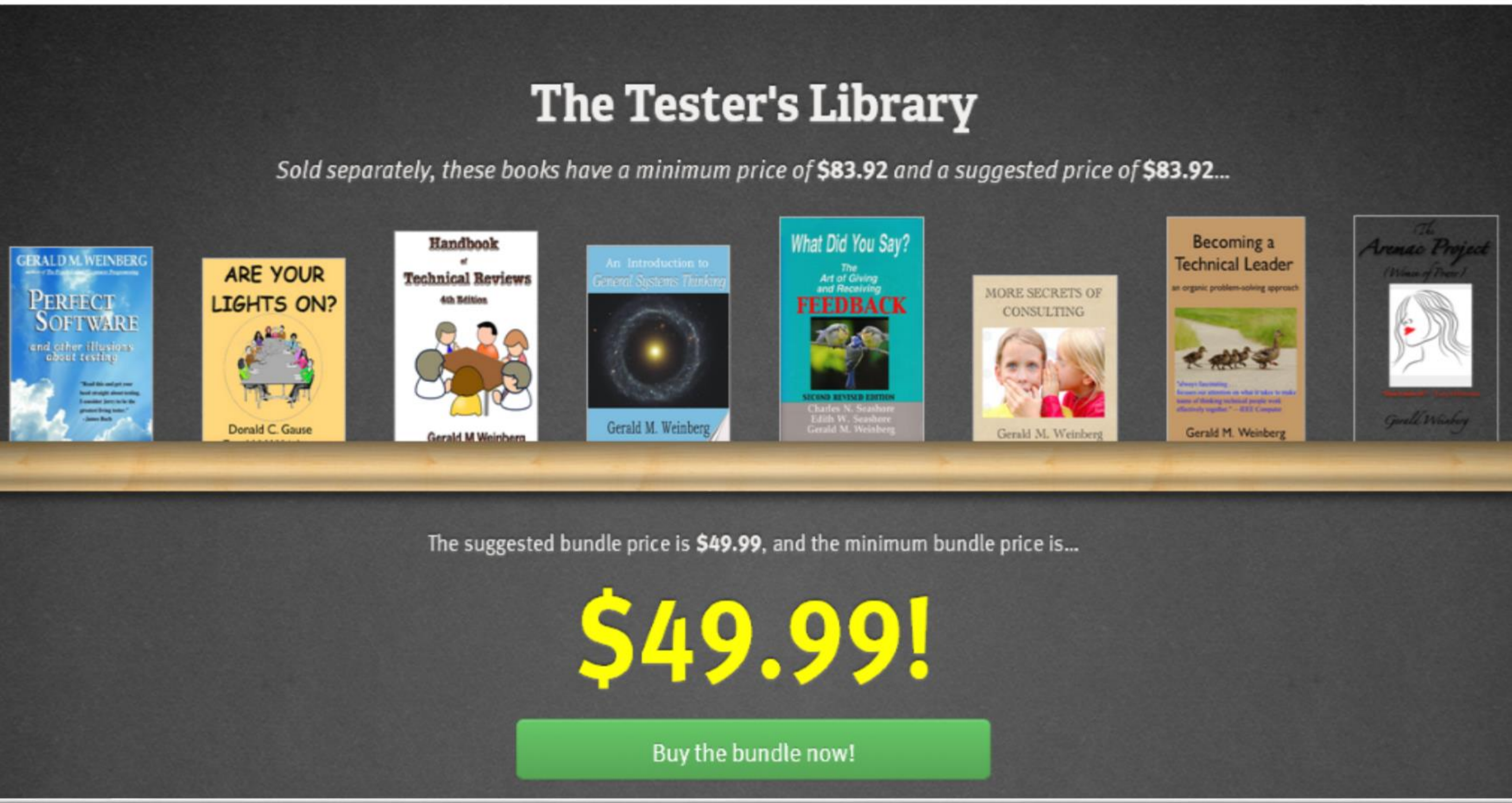
Lately, he's been observing the Agile movement, and he's offering an evolving set of impressions of where it came from, where it is now, and where it's going. If you are doing things Agile way or are curious about it, this book is for you.

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

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...

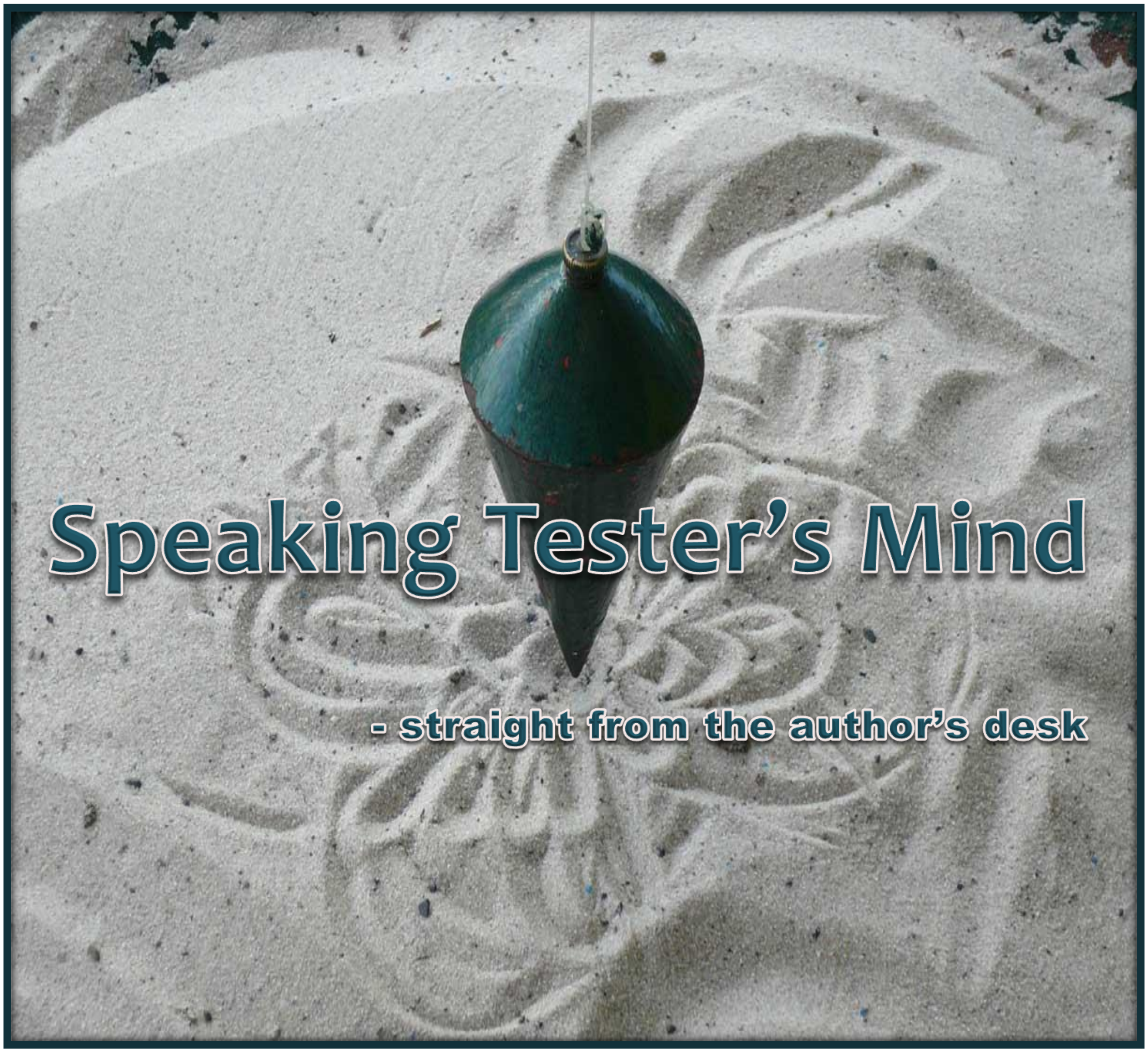
$49.99!

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


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

Testing Skills – part 4 Note Taking

- by John Stevenson

Why is note taking an important testing skill?

There are a variety of ways to capture the evidence of our testing but if are notes are not of suitable detail then the value of our testing can be diminished. Taking notes enables us to improve our knowledge and reinforces our understanding of the product being tested. This is part of utilizing critical thinking skills which was discussed in the chapter on 'critical thinking'

Robert Lambert discusses the need to have good note taking skills when performing exploratory testing

"During a session a good exploratory tester will often narrate the process; all the time making notes, observations and documenting the journey through the product. This level of note taking allows the tester to recall cause and effect, questions to ask and clues to follow up in further sessions."

Explaining Exploratory testing relies on good notes - Robert Lambert - 2013

Michael Bolton wrote the following about note taking when testing:

"One of the principal concerns of test managers and project managers with respect to exploratory testing is that it is fundamentally unaccountable or unmanageable. Yet police, doctors, pilots, lawyers and all kinds of skilled professions have learned to deal with problem of reporting unpredictable information in various forms by developing note-taking skills."

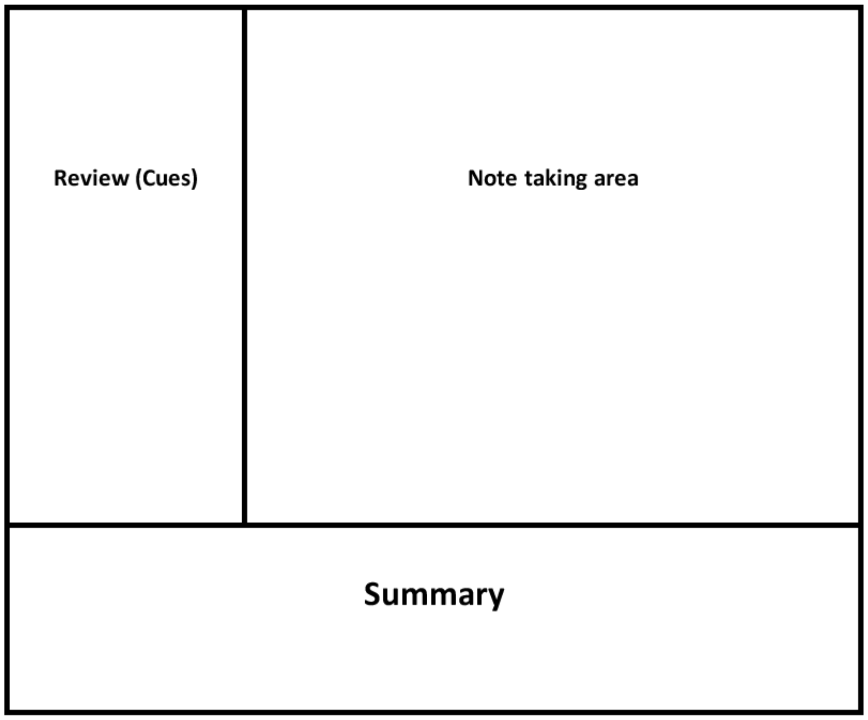
An Exploratory Tester's Notebook - Michael Bolton - 2007

There are a variety of note taking methods which you as a tester can utilize. This page has an example of a few of them. Note Taking Systems - Student Academic Services - Cal Poly

One method that I have found extremely useful especially when capturing information from conferences or for recording my findings when testing is the 'Cornell Method'

The Cornell method was developed by Dr Walter Pauk of Cornell University and is widely used by University students. It is a very useful method to help you work out if you can remember what you have written.

First of all you need to create a layout for each page in your notebook as follows. Alternatively use this Cornell Method PDF generator.



The method has 5 steps.

1. Capture what is being said or what you observe in the note taking area

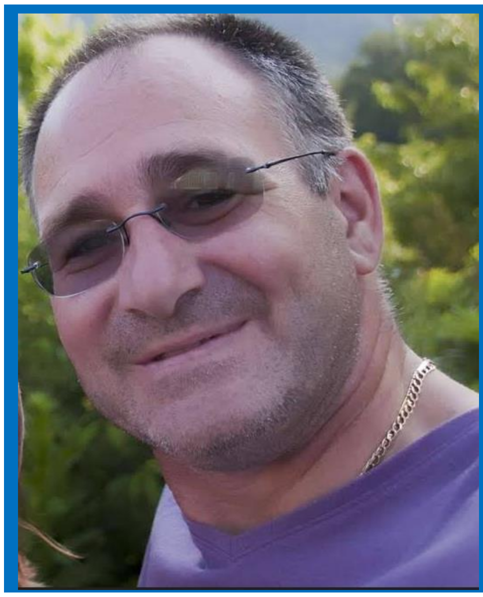
2. As soon as possible review your notes and capture key details in the Review (Cues) column, add any questions you may have thought of.

3. Cover up your notes only showing the review column and now try to summarize your thoughts based on the cues. Provide answers to any of the questions you wrote in the review column. Use the summary column at the bottom to summarize your understanding and learning. If you are struggling with your summary it could indicate that your notes are not sufficient.

4. Ask yourself questions on the material both the cues and the notes. Think about how you apply this information to your work. How does it fit with what you already know?

5. Spend some time reviewing your notes and summary every so often to reinforce your understandings.

It is crucial that as a tester you practice your note taking skills. Poor note taking can lead to missed problems and hinder knowledge sharing with the team. Your notes are what helps to turn your tacit knowledge into explicit knowledge.



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.

Back To Index

The future of software testing: How to adapt and remain relevant

- by Matt Heusser

Does having human testers slow down the performance of a software development group at all? That's what Yahoo seemed to claim in a recent IEEE Spectrum article that discussed how it eliminated the tester role from the organization. Its primary claim was the following:

"What happens when you take away the quality assurance team (QA) in a software development operation? Fewer, not more errors, along with a vastly quicker development cycle."

While the IEEE article claimed that having traditional testers is detrimental to team performance, you need to understand the full context. While traditional software testing roles in QA teams may be going away in some organizations, the work testers do isn't. If you understand what changes are coming, why it's happening, and how to hone your skills and adapt, you'll not only survive but thrive in this new environment.

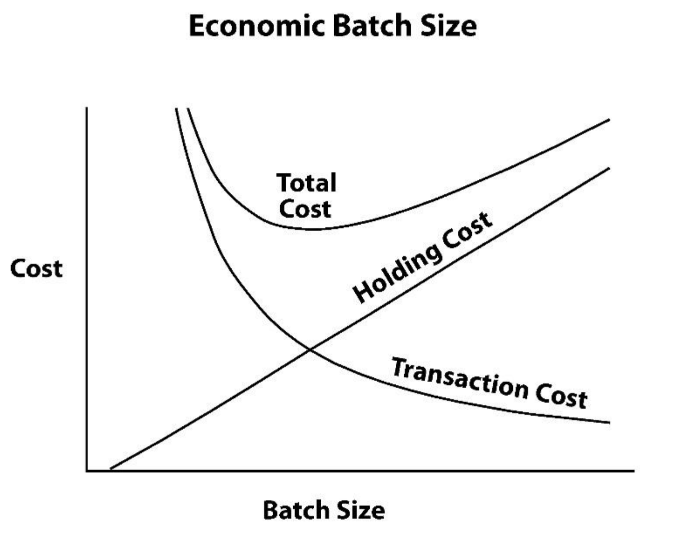
The problem with traditional testing groups

The process described in the IEEE article involves a development group handing off work to a testing group, with both groups primarily coordinating through paperwork and bug tickets. This situation creates a delay in the find/fix/retest loop. If the programmers have a high level of work in progress, there can be a delay between find and fix. Likewise, if testers have a great deal of work to test, there could be another delay between fix and retest. Written communication allows only slow feedback, which leads to arguments about whether the issue is actually a bug, should be fixed, works on the developer's machine, and so on. The article implies that the testing work involved retesting of the same thing over and over, following the same steps. This is sometimes called scripted/manual regression testing.

About the time that the extreme programming trend took off, Elisabeth Hendrickson was demonstrating the risk associated with a multi-team model. In her 2001 Software Measurement Conference presentation, "Better Testing—Worse Quality?" she suggested that programmers who knew that someone else would be checking their work would be less likely to check their own work, leading to lots of bug filing, fixing, and retesting.

Due to poor coding practices, this could lead to more bugs, repeating the cycle. Hendrickson's work included a systems effect diagram showing that some of those consequences are natural, while others happen due to a management choice. For example, management, driven by deadlines and only looking at "code complete," may push programmers to go faster. Forced to steal velocity, programmers may then do shoddy work and skip testing. Why not skip it, the thinking goes, because isn't that what the test department is for?

That's not the only problem with having a traditional test department.



From "The Principles of Product Development Flow," by Donald G. Reinertsen. Celeritas Publishing: 2009. Copyright 2009, Donald G. Reinertsen

Batches, delays, handoffs, and touch time

Don Reinertsen's book, *Principles of Product Development Flow*, discusses two competing issues: transaction costs, which are the costs to get any piece of software released, and holding costs, which are the economic costs of not releasing. For physical goods, this includes things like the cost of warehouse space; for software, it's the cost of delaying a product—the opportunity cost that could be realized if the software were just out.

According to Reinertsen, the rational choice is to find a sweet spot in the intersection of these costs and to release less often, reducing the number of transactions per year, but not so rarely that you never make any money.

If testing takes three months, you aren't going to release every day. (If you did try to release every day, you'd get just under four releases a year, each of which has one day of new features.) Instead, you'll release maybe twice a year and spend half your budget on testing. An executive with a new feature request might get the feature in four months, if they ask at the right time and can jump to the front of the line. Most likely, however, the executive needs to get the request in during planning, before the development cycle starts, making the delay from request to production more like seven months.

Knowing that there is a huge delay from idea to production, executives strive to get their ideas on the list as soon as they come up with them, making the backlog grow further. Because releases and patches are expensive, the test team wants to get the release right, so the duration of the test process increases, the transaction cost goes up, and it makes sense to release less often, with bigger batches of work. Bigger batches mean more unverified decisions, which mean more errors, which need more fixes, which take more time...and the cycle continues.

A few years ago, I had a client that spent about half of their time in a massive test/fix cycle. We talked about planning windows for meetings, saying things like, "It will have to be before August or after October." All of that drives up costs, but it is not the only way to think about delivery.

Let's go back to that example with the three-month release candidate testing. Say we really only did one day of development, that the release was thematic and hit only one subsystem, and that we had a way of isolating that subsystem so the changes did not damage other systems. Imagine further that you had techniques to monitor production and could easily roll the change back. Would you really spend three months release-candidate testing everything? Probably not.

The traditional, large-batch testing can sow the seeds of its own destruction, not just from a quality view, as Hendrickson pointed out, but also from an economic perspective.

New model: Testing and continuous delivery

The ideas above, taken to an extreme, point to the continuous delivery that Yahoo is advocating. Automated tools can take any code change, perform a build, and push those changes to a personal or staging server. That does not mean that the code automatically goes to production, a process which could be called continual deployment. Instead, a workflow engine can notify someone that the work is ready, and that person can then test that new piece of work. As Carmen DeArdo pointed out during the recent DevOps Enterprise Summit, the delivery model (continuous or traditional) does not matter, as long as delivery is consistently managed through the same tool. That means some teams can go to continuous delivery, while others release weekly, and regulated or waterfall projects release twice a year, without sacrificing the quality of information for decision-makers.

What should you make of all this? First of all, understand that companies don't shift to DevOps overnight, and even if they did, testing work would remain. Someone has to explore the software to find bugs and reduce risk. The ability to isolate a change into smaller and smaller batches, making release cheap, and the ability to roll back make it tempting to give the entire delivery, end-to-end, to a pair of programmers.

Testing still happens; it is just not a specialty role on the team. Eliminating the test "handoff" removes a delay from the process and speeds it up.

Is testing dead?

It has been four years since James Whittaker, then a director at Google (now Microsoft), stood up at a STARWest event and proclaimed that testing was dead. There has been a great deal of rhetoric, including Google directors dressed up as the grim reaper. Now that Microsoft and Yahoo have eliminated traditional test positions, the current state of rhetoric is that automation is the future.

While these rumors of the death of software testing persist, they're greatly exaggerated.

What might change is the easy, repeat-the-steps, follow-the-process testing jobs supported by a test organization with its own managers, directors, and vice presidents of software quality. What becomes of these testers? Some might become coaches or "smoke jumpers" who contribute to projects in need. Overall, however, the ratio of testers to developers will continue to shrink and the average skill level required will be higher. In some organizations, traditional testing jobs may become exceptionally rare and may go by different titles.

The question for testers today is not *if* the role will exist; it's if testers are willing to make the investment in remaining relevant. For example, the ability to spot problems in production, risks in requirements, or a combination of changes that could destabilize the system are all critical.

The new generation of testers must adapt their strategy to find emergent risks. That is, instead of running classic "check everything" regression-test processes, testers need to find out what is different and what matters for the release. That could include:

- Checking version control for what actually changed in the release
- Talking to developers about their concerns
- Studying production logs for what features customers are using
- Taking stock of the system's performance in production and what features are degrading over time

The point is to create a custom test process, one that is powerful, doesn't slow development, and provides feedback early. It should provide feedback not only about the code, but also about all the stages of the process. That includes fleshing out the details on features to reduce defects on the first build, and identifying the unintended consequences of requirements.

That's what you need to do to survive—and thrive—as a tester in this changing climate. So buckle up, but try to enjoy the ride.

Note: This article has first appeared on TechBeacon.com. We'd like to thank Matt and Robert Mitchell for helping us reprint it for our readers.

As the Managing Director of Excelon Development, **Matt Heusser**, consults, trains, and does software delivery while helping others do it. Probably best known for his writing, Matt is the lead editor of "How to Reduce The Cost of Software Testing" (Taylor and Francis, 2011), editor for Stickyminds.com, and recipient of the 2015 Most Popular Online Contributor to Agile at the Agile Awards. A 2014 recipient of the Most Influential Agile Test Professional Person Award (MAITPP) in Potsdam, Germany, Matt also served as the lead organizer of the Software Testing World Cup.

He is a former member of the board of directors of the Association for Software Testing and creator of Lean Software Testing family of methods.

jamo solutions

How to reduce the cost of Mobile Application Testing

REGISTER NOW

People at Jamo Solutions have asked Matt to lead a Webinar on reducing the cost of mobile testing - https://lnkd.in/e6kigQT based on the work he and other authors did on the "How To Reduce The Cost Of Software Testing" book (2011, Auerbach Publications) and what Matt has learned since. His part of the session is pure dynamics of mobile testing, risk management, and a tech techniques to reduce time to market without reducing risk. After his presentation the team from Jamo will spend a few minutes talking about some of their mobile test tools. We hope you'll join them.

Love to Write?

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

Send your articles to editor@teatimewithtesters.com

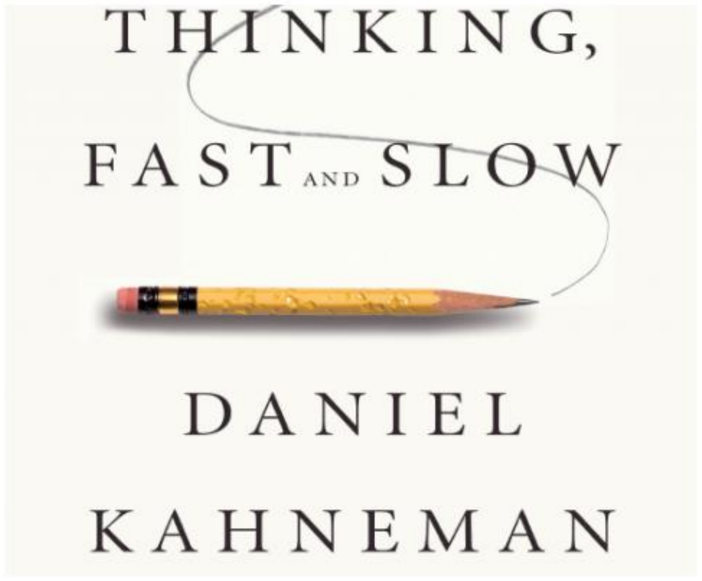
In the school of Testing
for your better learning & sharing experience

Mapping Biases to Testing – part 1

- by Maaike Brinkhof

We humans are weird. We think we can produce bug free software. We think we can plan projects. We think we can say "I've tested everything". But how is that possible when we are governed by biases in our thinking? We simply cannot think about everything in advance, although we like to convince ourselves that we can (confirmation bias).

In his book "Thinking, Fast and Slow", Daniel Kahneman explains the most common thinking biases and fallacies. I loved the book so much I've read it twice and I'll tell anyone who wants to listen to read it too. For me it is the best book I ever read on testing. That's right, a book that by itself has nothing to do with testing, taught me most about it. Before I read the book I wasn't aware of all the biases and fallacies that are out there. Sure, I noticed that projects always finished late and wondered why people were so big on planning when it never happened that way, but I didn't know why people kept believing in their excel sheets. In that sense, "Thinking, Fast and Slow" was a huge eye opener for me. There are lots of examples in the book that I answered incorrectly, proving that I'm just as gullible as the next person.



But that scared me, because it is my job to ‘test all the things’, right? But if my thinking is flawed, how can I possibly claim to be a good tester? I want to try to weed out as much of my thinking fallacies as I can. This is a journey that will never end. I want to take you with me on this journey, though. The goal is as always: improve as a tester. Enjoy the learning process and explore. I feel the need to put a disclaimer here. This is not a scientific type of blog series. I will provide sources where I think they’re necessary, but the point of this series is to take you on a journey that is for the most part personal. I hope it will benefit you as well! My goal is mainly to inspire you to take a look inwards, at your own biases.

Before we continue I need to explain a few basic concepts: fast and slow thinking, heuristics, biases and fallacies. I will conclude this first post with a list of the biases and fallacies that I will cover in this series. This list can grow of course, based on the feedback I hopefully will receive.

Fast and slow thinking

This is a concept taken from Kahneman’s book. Fast thinking, called “System 1 thinking” in the book, is the thinking you do on autopilot. When you drive your car and you see something happening, you react in the blink of an eye. It’s also the thinking you do when you meet someone new. In a split second you have judged this person based on stereotypes. It just happens! It’s fast, automatic, instinctive, and emotional. The system 1 thinking is the reason we are thriving today as a species (it helped us escape from dangerous situations, for example).

On the other hand, there’s “System 2 thinking”. This is the type of thinking that takes effort; it’s slow. It’s deliberate. For example, you use system 2 when you have to calculate (in your head) the answer to 234 x 33 (as opposed to 2 x 3, which you do with System 1).

There is one huge problem: we make all kinds of mistakes when it comes to using these systems. Sometimes, we use system 1 to analyse a problem, while system 2 would be more appropriate. In the context of testing: when someone comes up to you and says “is testing finished yet?”, you might be tempted to answer “no” or “yes”, while this is more a type of question that cannot be answered with a yes or no. If you want to be obnoxious you can say testing is never finished, but a more realistic conversation about this topic would be based around risk, in my opinion.

Often, when people ask a seemingly simple or short question, such as "is testing finished yet?", they mean something different entirely. In my context, if the Product Owner would ask me “is testing finished yet?”, for me it translates to: “do you think the quality of our product is good enough to be released? Did we build the thing right? Did we build the right thing? I value your advice in this matter, because I’m uncertain of it myself”. But if I happen to be in a foul mood, I have an option to just say "yes", and that would have been my system 1 answering.

Putting in the mental effort to understand that a simple question can actually be about something else, asking questions to find out what the other person truly means and crafting your answer to really help them, is hard work. Therefore, you have to spend your energy wisely.

Develop your system 1 and system 2 senses: when do you use which system? And then there’s the matter of choice. It would be silly to think you can always choose which system you use.

That brings us to heuristics.

Heuristics

Definition on Wikipedia: "A heuristic technique, often called simply a heuristic, is any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals."

Heuristics are powerful, but you need to spend time reevaluating and/or adapting them once in a while, and for that you need system 2. Why do you need to reevaluate your heuristics? Because you are prone to fall for biases and fallacies.

We need to use heuristics, but they are based on system 1. When you are an experienced tester, you have a huge toolbox of heuristics that help you during testing. That's a good thing, but it comes with a risk. You might start to trust your heuristic judgement a little too much, but you can't use a hammer for everything, right?

Bias and Fallacy, definition and meaning

A bias "is an inclination or outlook to present or hold a partial perspective, often accompanied by a refusal to consider the possible merits of alternative points of view."

A fallacy "is the use of invalid or otherwise faulty reasoning, or "wrong moves" in the construction of an argument."

Most of the thinking errors I will cover in this series are biases, but it is good to know the difference between fallacy and bias. A bias involves a mindset; you see something in a pre-conceived way. It influences how you experience things. Stereotyping is a common example of a bias. A fallacy, on the other hand, is an untruth. It is a statement or belief that lacks truthfulness.

There is more than one type of bias, but in this blog series I will talk about cognitive biases, for which the definition is "[...] a repeating or basic misstep in thinking, assessing, recollecting, or other cognitive processes."

Since testing is a profession that relies heavily on cognition, mental judgement and we are only human, it's no wonder that we make mistakes. You cannot get rid of all your biases, that would defy human nature, but in the context of testing it's a great idea to challenge yourself: which biases and fallacies are actually hurting my testing activities?

However, you have to realise that biases can work to your advantage as well! Since it is part of our human nature to be biased, we should use that fact. With regards to testing that could mean: get more people to do testing. Every person brings his or her unique perspective (with biases) to the table and this will result in more information about the application under test.

What’s next?

In this article series I hope to shed some light on a number of biases and fallacies and what harm or good they can do in testing. I will cover the following biases, fallacies and effects:

- anchoring effect
- availability heuristic
- framing
- sunk cost fallacy
- cognitive ease
- confirmation bias
- priming
- hindsight bias
- attribution bias
- *"What You See Is All There Is"*

If you have more input for biases or fallacies that you want to see covered, please leave me a tweet @Maaikees

Maaike Brinkhof - Test Consultant at Xebia Netherlands

Maaike is an agile tester. She loves testing because there are so many ways to add value to a team, be it by thinking critically about the product, working on the team dynamics, working to clarify the specs and testability of the product, getting the whole team to test with Exploratory Testing…the options are almost endless! She likes to help teams who are not sure where or what to test.

After reading “Thinking, Fast and Slow” by Danial Kahneman she developed a special interest in biases with regards to testing. During ‘analogue time’ Maaike likes to practice yoga, go for a run, check out new local beers, play her clarinet and travel with her boyfriend.

Back To Index

Waterfall to Agile – A Test Team's Unexpected Journey

- by Sakis Ladopoulos

Introduction

This is the story of a test team's unexpected journey from Waterfall to Agile and a few steps back, finding a middle ground, their middle earth, in a proprietary test shire. The story how independent testing diminished close to become obsolete but rise up again the very last moment with a new formation. It's the story of a team of testers lost in agile translation who found D.A.D. (Disciplined Agile Delivery that is) and through D.A.D. a new refined identity for them. It is my personal story as a test manager how I almost lost my job, having my role threatened with extinction and then turned out having every test manager's dream come true: insight within development activities their progress and their quality and a robust way to forecast how much testing is needed according to the actual quality delivered to test team even before delivered. Last but foremost it is an important experience of how a software development corporation answered the most fundamental question when it comes to organize functions: *United or Decoupled?* A question that encrypts the same underlying dilemma only seen from methodological perspective when questioning: *Waterfall or Agile?*

Living in the Waterfalls

Once upon a time there was a software house with an independent test team which was following the old classic software development lifecycle methodology: *waterfall*. V-model was ruling all development activities. Analysis, development and testing were three distinguished functions staffed through different teams and everyone was expecting the previous stage of work to finish so as to receive input and proceed with the next step in a totally sequential manner like an industrial production line.

This software house was generally well reputed as it was delivering on time and with quality at least in most of the cases and most of the time. The bureaucratic way of work though was not leaving any flexibility to deal with scope changes or changes in general. Well-structured and clearly defined projects where treated significantly well. Projects within dynamic environments and continuous changes where considered a nightmare that cannot and should not be handled by well reputed organizations and should be just avoided. In the unlikely event that you get stuck in such a project the best thing considered was to cancel it as soon as possible, to burn it down with the minimum cost for the software-house.

Due to that, this software house was in fond of stable customers and stable customers do liked this software house. Most of the projects were contracts with National Administrations and the Public Sector. Our software house was really happy with these large framework contracts from the NAs even though the cumbersome way of work never actually made it to leave a considerable contribution margin.

Sometimes the strict and bureaucratic processes were creating delays or gaps in final outcome and in regards to initial request but that's life, there was no other way considered. So we were all happy in our little shire even though we were struggling with major constraints and shortcomings. This was all we knew. This was what we knew that we could do.

The ring

One usual, not so shiny, day a big and unexpected announcement for an imposed organizational transformation was made. The storm of an unmanaged change with uncalculated risks came to our shire to rule us all. Who was the bearer of it; it is of little importance. The change, which I will call the ring in our case, was called *Agile processes*.

The bearer was dazzled by the power of the precious new processes which seemed to solve every problem. The bearer said that we should leave behind the old fashioned, bureaucratic and cumbersome procedures that we knew and ride the wave of the new era of methodologies. Methodologies who promise flexibility and adaptability, who let you deal with unforeseen changes. The bearer said nothing about the hazard of naïve use of the ring. Probably even the bearer could not understand that the powers of the ring could destroy even him.

From a moment to the next all that we were following was treated as cursed and transformed to the scapegoat for all evil. Many had reservations for the imposed transformation, some had great concerns and I was fighting back as much as I could. To my great frustration and disappointment I soon realized that for a decision, already taken, coming with a top-down approach, there was no room for different views and no one was willing to hear for risks and pitfalls. Trying to express my concerns I even once got the answer: "Don't worry you will not get disappeared. You could easily work as a scrum master in the new formation"

All the ones having opposing views were desperate. We looked like and treated like we were opposing our corporation's mission and vision. We were practically in a battle already lost. Some were trying to see how they would fit in their new modern and funky agile costumes. Some were getting ready to leave along with their old fashioned three piece suits. Everyone was trying to live with it either taking it or leaving it.

"Gollum has some part to play yet, for good or ill"

At the point where no return was visible, an unforeseen ally suddenly appeared. One who erroneously enough was not initially considered as a stakeholder to this change but turned out to have a major role and an empowered say over the story, over the test team's unexpected journey. This was our dear and beloved set of existing customers.

It seems that over the years we are chosen by customers and we choose customers who work in a similar manner, who have the same consideration of how things work; who are driven by the same culture and ethics with us. We choose customers, partners in general, who look and feel like us, even though we thing of them as being *Gollums* when things in a partnership are not going well. Our customers liked us due to how we work and not only for what we deliver. They were in fond of us exactly for this cumbersome and bureaucratic way of work that we were following since they were following it too; the very same one that we were trying now to completely change.

Apparently they didn't like the forthcoming change and they started expressing the same anxiety with these old dogs in the software house who were opposing the idea of the change, who were afraid of the ring.

AGILE but … one does not simply walk into Mordor

This was enough to reconsider the decision taken, as customer is always right. Only then the change was envisioned not as an order to follow but as a request for proposal expecting ideas for implementation where constraints were acknowledged a priori. The most apparent constraint was that any change should not affect current ongoing activities, contractual obligations, interfaces or communications with customer. In other words any change should be transparent to the customer leaving intact current ongoing business.

After several brainstorms, reviews, discussions, agreements and disagreements a solution seemed to have been formed which was following the generally accepted facts that:

- Agile methodologies and specifically scrum seem to produce better results in development level. Scrum provides the required level of flexibility and adaptability to continuous changes and shifting needs during development
- An agile methodology on its own is naked. It cannot provide the necessary interface to the customer when customer is unwilling to follow agile and act as the product owner.
- It does not provide a suitable procedure to frame the overall work from its initiation neither can set clear transition rules, criteria and expectations.

The solution was a hybrid schema where scrum was followed as Development process but the Development process was protected and encapsulated from a Service Delivery Process which was making in return the development process more or less transparent. All the interfacing with the customer was treated in the Service Delivery level. Service delivery level was acting as the adaptor between the customer processes and our software development plant. It also provided the opportunity to scale agile development through forking a project to several scrum teams.

This pioneering idea was thrown before a couple of years, only a few months after the release of Disciplined Agile Delivery which is practically describing more or less the same concept. We actually do re-invented the wheel but at least that was a good indication that we were on the right course. DAD extends the construction-focused lifecycle of Scrum to address the full, end-to-end delivery lifecycle from project initiation all the way to delivering the solution to its end users. According to DAD the construction phase which can be either scrum or any other agile process is encapsulated by an Inception process in the beginning of the project and a Transition process in the end of the project. This hybrid model helps to stream line all IT related work and provides a solid foundation to scale it.

What all these mean for our test shire?

In waterfall processes testing is a separate phase before the delivery of the project. For a software house following waterfall, test is a different team/ function under its own management following their own testing processes and methodologies.

In agile processes we have exactly the opposite. Testing is a continuous parallel activity bundled together with all other activities and part of work of each and every cross-functional engineer. Testers and test teams have no meaning at least in theory of agile processes.

What was the future, if any, of a test team in this hybrid formation?

Test Team not only didn't seize to exist but turned out to have a leading role in the implementation of the hybrid model. Test Team now provides services as an Independent test team along with the provision of Test engineers if and whenever needed in scrum teams. Test Team now glues the construction phase implemented by scrum teams with the transition phase which streamlines the processes and products towards to a customer delivery model which was left intact.

Test Team was kept as one function under its own management. Yet the team is providing services in two totally different operational models. A part of the team was utilized within scrum teams, usually one engineer to every scrum team, having a more testing flavor cross-functional role. Another part was pretty much organized as it was previously as an independent test team.

Team's management was responsible for the resource allocation of both parts and a critical factor of the resource allocation was to ensure that there is continuous circulation of the testers between the two parts. No one was left for too much time in a scrum team avoiding to be completely absorbed by the cross-functional profile and every member of the independent test team was expected to take part in a scrum at a specific point of time.

Following this structure of the test team we now have:

- Better visibility of the incoming package
- Smoother internal transition from DEV to TEST
- Greatest variety of test positions
- Better technical visibility and awareness of activities before testing

… And all live happily ever after? - Lessons learned

The described organizational transformation turn out to be really successful but this is just the story of a specific software house with its unique needs and peculiarities. The most important part from a transformation endeavor is the lessons learned which can be summarized below and kept as a moral of the story …

- Go with the flow but beware of the trends especially if you are trying to always be an early adopter. They can wash you away.

Processes and methodologies follow a cycle of born – hyped – used – misused – fought – burned. This is the lucky story of every new industry trend and only after some time this cycle ends we can actually evaluate the efficiency of any new idea or approach, model or process. This happens because we tend to adopt in an impulsive and unrated way every new big thing. Trying always to be on the edge and not to be considered obsolete following yesterday's news we ride on the wave of every new technology or methodology without really evaluating if it fits for purpose or not. Scrum is out on the field for 20 years now. Still there are way too many software houses that are working with waterfall processes, sometimes ashamed to admit it, and they see no reason to change that. There must be a reason for that!

- Do not re-invent the wheel (as we did) but there is no top off the shelf product that will take care of your specificities and peculiarities.

Rarely something off-the-shelf, broadcasted for everyone proves to be the perfect solution, the right answer for our unique needs and specificities. New industry trends should be observed as interesting innovations and fresh food for thought but should not be followed as the one solution to all of our problems. We should always try to find our own unique solution for our own unique business problems or our own unique way to exploit our own unique business opportunities with our own unique resources we have. As said, one does not simply walk into Mordor …

Sakis Ladopoulos is a Test Manager with substantial experience in forming, leading and managing through changes, teams of test engineers in IT and Telecom. Apart from Software Testing, which was his first job in Siemens AG more than a decade ago, he has also worked as internal auditor for Quality Management Systems and as a member of several work groups and committees for ISO and CMMI certifications, having gained that way a full oversight of Quality Assurance & Control area within Telecom and IT industry.

He is a frequent writer and speaker in software testing related topics and a proud father of two little "stress testers".

Back To Index

BRAND NEW SHOW: Techno Talks with Lalit on www.tvfortesters.com

Episode 1 – coming soon

Talking Mobile App Testing with Testing with Daniel Knott

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

Becoming a Testing Craftsman...fine aspects to mastery

As we mature, it is not just about doing work well, it is about it so well that it exudes a touch of artistry. That is the mark of a craftsman, one whose work is seen as one-of-kind, an object-de-art. This is what distinguishes one from the rest of the pack. It is about fine aspects, the subtle things that make a huge difference to the quality of work and therefore the value we offer to our customers.

What do we see in a fine craftsman?

Craftsman at work

The passion they exude, the sheer joy they display at work, the minutest attention to details, the elegance with they use the tools, the clear mental image of outcomes they visualise, a well thought out plan that evolves continuously. With minimal information given on what it is be done/expected, they dig for information, discover missing pieces weaving them to create the full picture. They approximate, iterating multiple times to get to the required degree of precision. Measuring with precision and doing refinements, keeping the big picture even while working at the lowest level. Whey they work, demanding and complicated tasks seem to be done effortlessly. Seeing a fine craftsman at work is just lovely.

Craftsman as an individual

As an individual they take extreme pride in their work and good work is their ultimate reward. They are their own critique, judge their work carefully and not swayed by praise/criticisms from others. Appreciation from another craftsman is valued deeply whilst criticism is humbly accepted. They absorb knowledge from a multitude of sources and always willing to learn. Humility is a key trait. They are not afraid to try new things and tread into new territories after deep study.

Craftsman as teacher

A good craftsman does not teach, rather they enable discovery. They do not teach, they mentor/groom. They value observant students, are very demanding in the work they expect of them. They device simple exercises and expect the student to practice it umpteen times until they get it perfect. The tolerance to casualness in approach and shoddy outcomes is very low with the student being shown the door. The student outshining the master is the ultimate tribute to the master.

What are some of the fine aspects to mastery?

Passion, pride and fine attention to details

Passion in what we do is what compels us to excel and strive towards perfection. The pride in ensuring the finest quality goes beyond the final outcome of testing, it is about ensuring that all activities of testing are done very well. It is not just the outcome of testing being effective & efficient, it is about paying attention to every detail in the activities performed, be it setting up a strategy, devising a plan, designing test cases, developing scripts or sending reports.

In the case of strategy/plan, it is about being crisp, purposeful, being 'non-jargonish', thinking through every small detail enabling clear action plan. In the case of test design, writing less and communicating more by clear and crisp documentation, using a consistent language syntax pattern to describe scenarios that enables scenarios/cases to be related easily to real life usage. In the case of reporting, it is about providing a clear bird's eye view first, enabling rapid understanding-in-the-large, then swooping down to each key element and providing the fine levels of details as appropriate.

Clear mental imagery & visualization

Every system is deemed complex until simplified. Visualisation of the system as a set of business operations done by end users which are decomposed into technical features delivered a composition of building blocks/components enables a tester to be unfazed by the complexity of the system features/behaviour. Visualisation the internal structure as to what the various components are, how they are interconnected, and the flow of data enables a good imagery of the how the system is/will-be made up of. Complementing the above static views of internal structure and external behaviour with the usage enables a clear mental image of who uses and how-much, enabling the visualisation of the system in operation. Striving for this degree of clarity enables the tester to dig for information by questioning deeply or suggest/advise to fill-in the missing information.

Elegance in using tools

The act of creation requires tools, the craftsman uses these tools with grace and elegance as he transforms the idea into a real product. Test design denotes this creational aspect in testing where we create test scenarios/cases (& later scripts) that digs into the system to uncover defects. The tools in the act of test design are the various techniques that we use to fashion out test cases. Using appropriate design techniques to come up with elegant behavioral models that are clear, simple, well segregated into different behaviour categories (like functionality, performance) makes this creational (design) process elegant resulting in a well-organized set of effective scenarios/cases. Improper use/composition of design tools invariably results in a 'spaghetti' test cases that don't display grace or elegance.

As we continue our testing journey, we strive to become better, to become a craftsman. This journey is a lifelong pursuit, the reward being the good work done. Here in this article, I have attempted to list out some of the fine aspects to mastery of test craftsmanship with the next article moving onto the individual traits craftsman in the context of testing.

Ask yourself as where you are in this journey, list down work products, activities that you are proud of, work products that have your stamp of unique signature.

Enjoy the good work, produce object-de-art.

Until next time, CIAO.

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

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

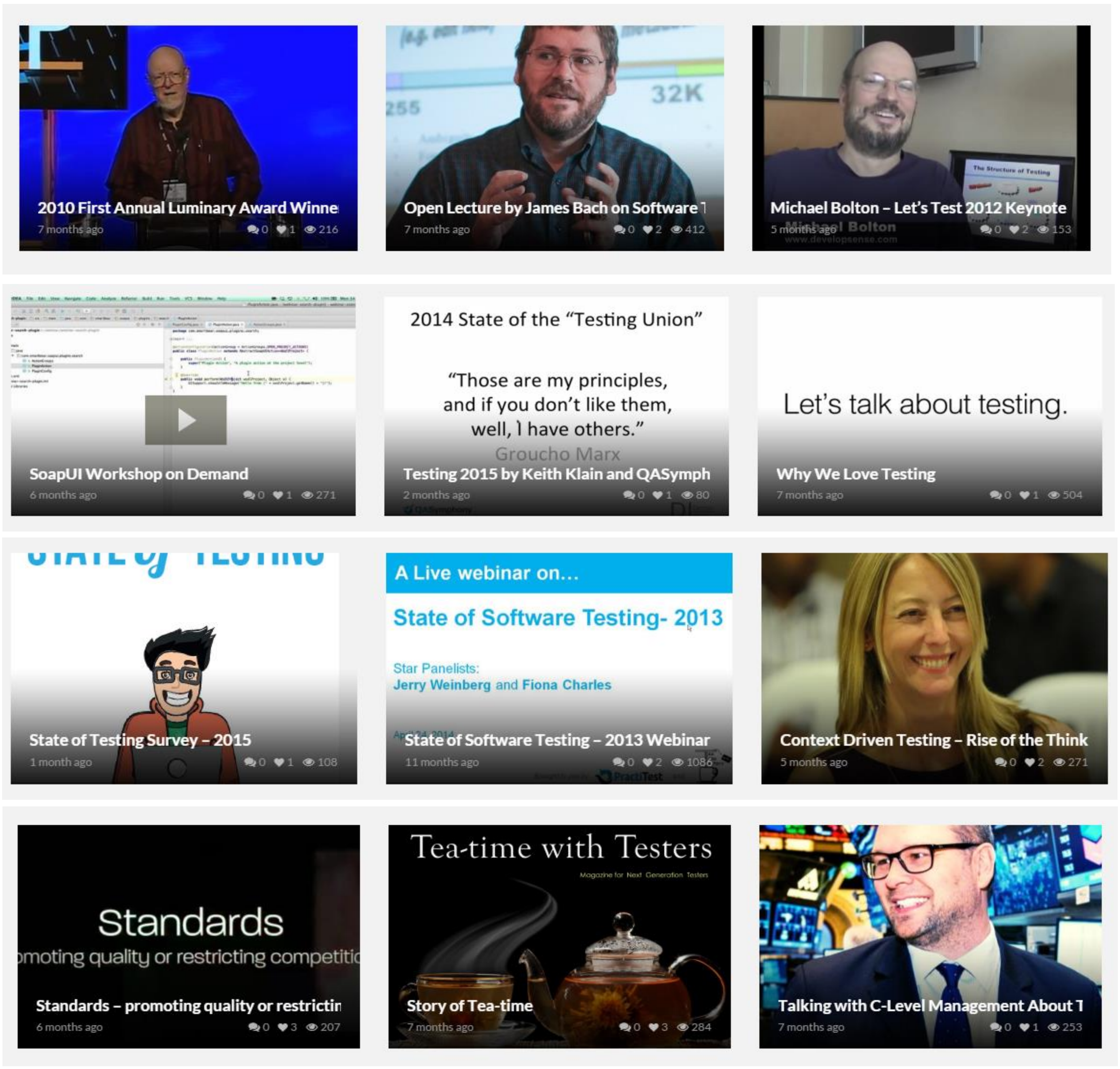
He can be reached at **ash@stagsoftware.com**

Back To Index

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

TV for Testers

Your one stop shop for all software testing videos



2010 First Annual Luminary Award Winne
7 months ago 0 1 216

Open Lecture by James Bach on Software T
7 months ago 0 2 412

Michael Bolton – Let's Test 2012 Keynote
5 months ago 0 2 153

SoapUI Workshop on Demand
6 months ago 0 1 271

2014 State of the "Testing Union"

"Those are my principles, and if you don't like them, well, I have others."
Groucho Marx

Testing 2015 by Keith Klain and QASymph
2 months ago 0 1 80

Let's talk about testing.

Why We Love Testing
7 months ago 0 1 504

State of Testing Survey – 2015
1 month ago 0 1 108

A Live webinar on…

State of Software Testing- 2013

Star Panelists:
Jerry Weinberg and Fiona Charles

State of Software Testing – 2013 Webinar
11 months ago 0 2 1086

Context Driven Testing – Rise of the Think
5 months ago 0 2 271

Standards
promoting quality or restricting competitio

Standards – promoting quality or restrictin
6 months ago 0 3 207

Tea-time with Testers
Magazine for Next Generation Testers

Story of Tea-time
7 months ago 0 3 284

Talking with C-Level Management About T
7 months ago 0 1 253

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.

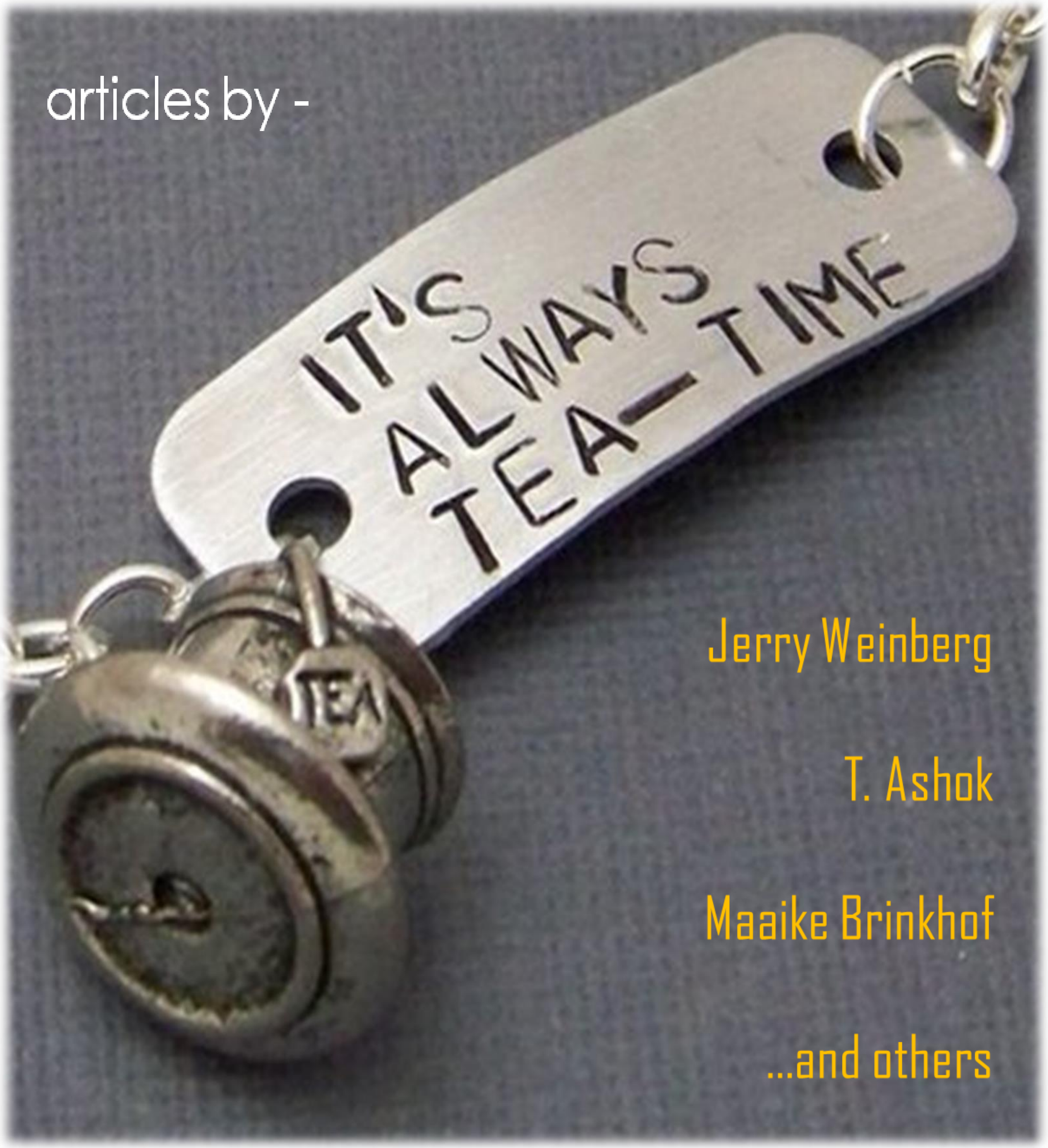
And the **Good News** is…

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 ?
Image : vernhart

in next issue



articles by -

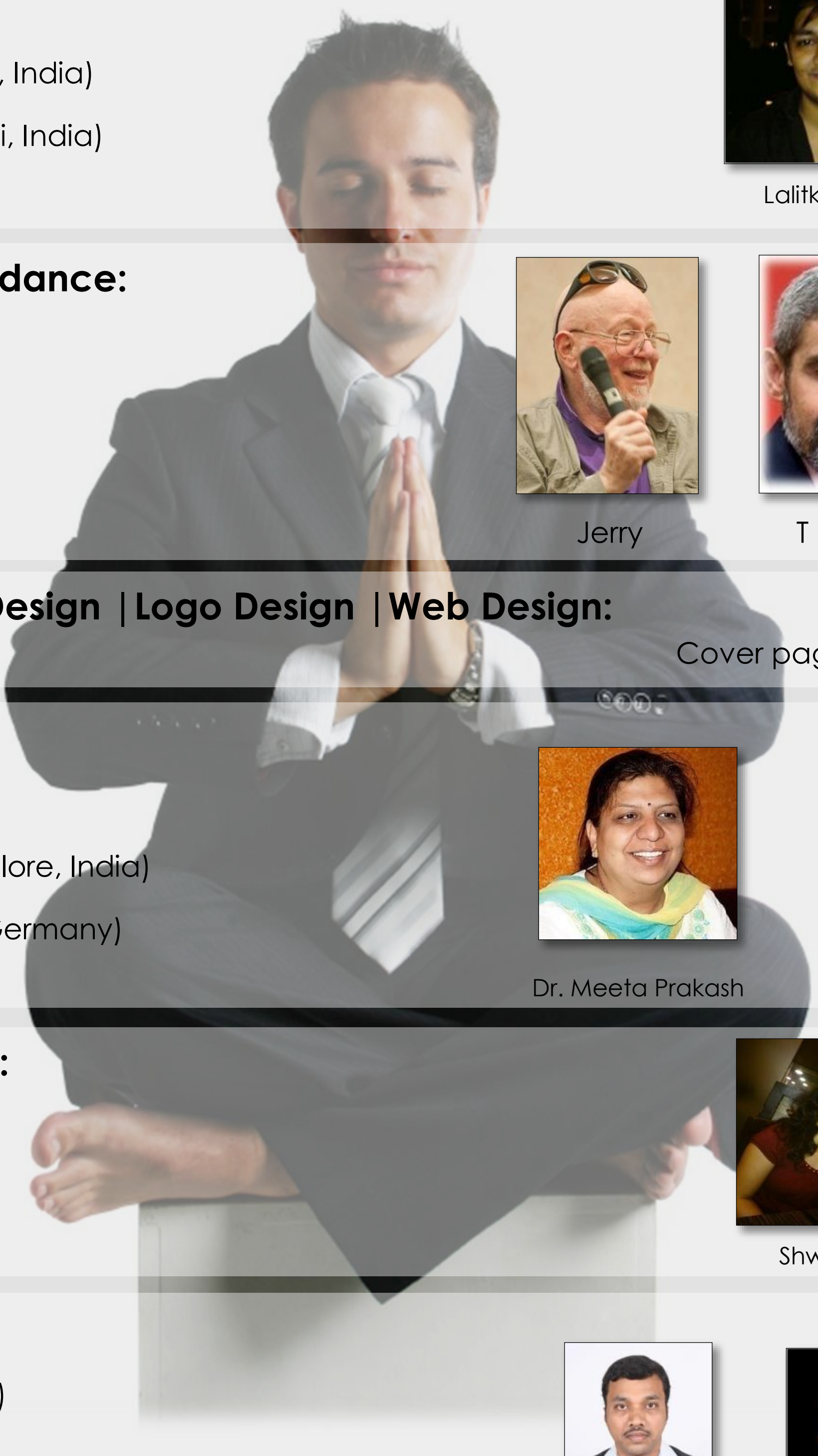
Jerry Weinberg

T. Ashok

Maaike Brinkhof

...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 – CDN Media

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

Give Feedback