THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

December – January 2016 | YEAR 5 ISSUE XI

Jerry Weinberg
Resting on Agile Laurels

John Stevenson
Testing Skills - Influencing by Listening

Shrini Kulkarni
The Promise of Small Tests

Albert Gareev
Do You See a Problem?

T. Ashok
Give up to Grow

Over a Cup of Tea with Dagmar Mathes

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

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

Editorial

Here’s to 5 awesome years!

As I am writing this at this very moment, I’m feeling nostalgic.

Somewhat around this time and day of the year 5 years back, I was writing editorial for first issue of Tea-time with Testers. Being a tester with merely 2 years of experience in the field, I was feeling bit nervous and excited at same time.

“With number of professional magazines already being there, will testing community welcome yet another publication?”, “Will the junior tester driving all this be taken seriously?”, “With no solid network in the community, how would I get articles for future issues?” were few among many questions running through my mind at the same time. But somewhere my heart was telling me that my ideas, efforts and intentions were genuine and that I must not step back just because of fear of rejection. Heart took over my mind and we launched first issue of Tea-time with Testers.

And the rest I believe is history.

Over these 5 years we did our best with what we thought would benefit testing community. We launched campaigns, encouraged and published thought provoking articles. Provided a platform for testers to voice their opinions, loud and clear. I firmly believe that Tea-time with Testers is the first Indian publication to do serious reporting on modern software testing theories and thoughts. And I promise to continue doing so.

Needless to mention, this was not possible without support of our guiding angels Jerry Weinberg, T Ashok, Joel Montvelisky and all authors who contributed quality articles for our publication, selflessly. Appreciation, love and support from you, the readers has always been overwhelming and I can’t thank you enough for that.

As we are stepping into 6th awesome year, I promise that you’ll get the best of what can do this year and for years to come.

Here’s to 5 years of awesomeness!

Sincerely Yours,

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

QuickLook

What's making News?

STATE OF TESTING -2016 Survey crosses 1000 mark!

Following in tradition of the past two years with the State of Testing survey, conducted in collaboration with **PractiTest**, we are happy to announce that we reached more than 1000 testers taking part in **State of Testing Survey 2016!**

Last year was very successful with almost 900 participants, thanks to all participants and collaborators because of whom we could hit our goal for this year.

The survey seeks to identify the existing characteristics, practices and challenges facing the testing community in hopes to shed light and provoke a fruitful discussion towards improvement. It would be interesting to see what 2016 has in store for testers and what the survey results reveal this time. We'll keep you posted.

You can download the survey result of last year from - **http://qablog.practitest.com/state-of-testing-2015-2/**

State of Testing 2016

More news? Visit Latest Software Testing News!

Meeting **Dagmar Mathes** has been an interesting experience.

My first interaction with Dagmar happened as a part of my interview with XING. Little did I know that a round scheduled for 45 minutes would turn into "some hours" of stimulating discussion where we discussed some interesting aspects of testing, challenged each other's ideas, agreed and agreed to disagree too. What won my heart was the seriousness with which a tester's hiring was being done and the absolute clarity of the panel members in terms of what they wanted from a tester. Meaningful interview is still a pain point for our community and that's where I decided to interview Dagmar for TTwT, regardless of the outcome of my own interview.

Dagmar has over 18 years of experience in testing field and worked for XING AG as Director of QA until recent. She is also one of the board members of German Testing Day conference.

Read on to know what Dagmar thinks about various aspects of software testing in this exclusive interview…

- Lalitkumar Bhamare

Over A Cup of Tea with Dagmar Mathes

Thanks for talking to us today, Dagmar. We are curious to know about your test side story.

Originally I studied geoscience and became a mineralogist. After two years working at the University of Hamburg I stepped into the IT and started in a small consulting company. After a few years working as a developer I got the opportunity to switch into the QA department of the customer I worked for. That was the beginning of my testing career.

Please tell us more about your work and experience at XING

At the end of 2007 XING decided to build up a QA team and asked the consulting company I worked for to help them. At that time I already decided to leave the company after in total 7 years but I was curious about the project at XING. I talked to both companies and started at the beginning as a consultant and switched to XING as an employee after the project was finished.

We started at the beginning of 2008 with one QA lead and 2 consultants in our headquarters in Hamburg. Today we are about 30 Software Test Engineers in Hamburg, Munich and Barcelona who are working in the agile development teams. Beside that there are 3 Test Architects who are responsible for the test infrastructure, test data provisioning and test automation and there is one Quality Manager who takes care of quality and test management across the whole company.

What are some testing skills that you value more while hiring new testers?

XING is using agile methodologies for about 8 years now. Since then QA is part of the development teams.

This means that we are looking for people who want to build a great product with a high quality together with the developers, product owner and designer. They focus on the product quality and not on processes, methods and tools.

Do you also think that role of testing in Software Development space is changing? How? And where do you see it leading to?

Yes, it's changing. When we started with the QA at XING we were looking for test analysts and test automation engineers. Today we need QA experts who can support the whole software development process. That means they cover different roles depending on the needs of the team they are working with. Some testers are focused on API testing, some on test automation; some are doing more mobile or manual testing. But they all can switch the teams easily and can step into the role which is needed.

It appears that in current market, there is increasing emphasis on test automation as compared to other ways of doing testing. What is your opinion about such shift?

Test automation is helpful, I agree. But a QA expert is doing much more than only writing test scripts. He is usually a tester who makes the current product quality visible, he is a product expert with a strong product and context knowledge, he is an open-minded thinker who challenges assumptions and decisions and gives constructive feedback and finally he is a coach and consultant with regards to quality topics.

XING appears to be a place full with lots of cool things and people. Would you like to share with us 'what testing at XING' means?

Yes, XING is a great company with great people. I would not say that we have any key characteristics of testing culture. We have a very open culture which means that we get the opportunities to try our new methods, processes and tools more or less whenever we like. We have every three months a hack week where all engineers (also test engineers!) get the chance to work on any project they are interested in. Once a year we have Prototyping Days where the whole company can participate and can bring up their ideas about new products, services or solutions. The QA colleagues are always very involved in these options and try out new test automation frameworks, methods or whatever they are interested in. Beside that we meet every week for our team meeting and share our knowledge. Everybody is involved in onboarding and training of new colleagues which means that we have a strong QA community at XING.

What do you think is the right way of measuring tester's performance?

Good question! We don't measure performance in the QA with any KPIs or something like that. We try to give everybody the opportunity to focus on topics beside their daily work in the agile teams. We have so called focus teams at XING which means that the products they are responsible for are in the focus during the year. Normally the tester in such a team is not able to do anything else than supporting his development team. Other colleagues are able to take over some more responsibility instead. This can be a bigger involvement in our QA goals, trainings, test support of the focus teams and so on.

Therefore it is not so easy to say colleague A has a stronger performance than colleague B. For us it is more important that the team delivers a high quality product. And of course we are proud if the team mates tell us that they don't want to miss their own tester.

For the better quality of product, if you are to change/remove/add one thing in Agile, what that would be?

I think in some cases it would be easier if the teams follow the same rules, processes and methods. In our company we have the agreement that every development team can decide more or less about the technology they want to use, the process they want to follow and the methods. This causes sometimes in an overhead of additional communication and finally in setting up processes for the collaboration with all the teams. As I mentioned before, XING is a company that always tries to develop and to improve itself.

You are on the conference board of German Testing Day. We are curious to know more about your work there…

I took over the responsibility after the former QA lead moved to Munich tree years ago. He recommended me as his substitute. In comparison to other conferences the German Testing Day is a non-commercial testing conference. Nearly all board members are working in a production company. We want to give everybody who is interested in testing and quality assurance the opportunity to join the conference and to share his experience. The conference is focused on practice and not on research or consultancy.

What you think needs to be changed in Software Development field (or testing field) with immediate effect?

I cannot say what needs to be changed in general. The companies I know have the challenge how to handle the faster time to market development. Due to that many companies think that an agile process is the only thing they have to implement and then everything goes well. They don't think about the people who have to make a switch as well. Very often they are told to read a book and the only thing they have to do is setting up a scrum wall and to organize their daily standups. That's not enough because they have to change their mindset as well. They need to understand that there are no separated roles anymore in a team. Everybody is responsible and everybody should be able to take over tasks of the colleagues in a small range.

According to you, how would software testing look like in next 5 years?

I think tester will become more coaches and consultants. If I look at the current situation at my company the testers are responsible for manual and automated testing for web and mobile in parallel. In average we have one to two testers in a development team. They cannot handle all this tasks in the same depth as before. They have to focus on the most important testing tasks which means that the developers and designers have to take over more testing tasks.

Last question for today. Do you read 'Tea-time with Testers'? We would love to get your feedback.

Yes, I do but not every article to be honest. I like the way you publish variety of articles very much, though.

Resting on Agile Laurels

"During a period of exciting discovery or progress there is no time to plan be perfect headquarters. The time or that comes later, when all the important work has been done. Perfection, we know, is finality; and finality is death." (C Northcote Parkinson).

Now that we're nearing the end of an exciting period of discovery and progress in the Agile approach, I thought you'd like to hear from a few of my former students on what it was like to become successful Agile teams. I was going to give you the five greatest success stories, but unfortunately, my headquarters aren't as well organized as I'd like.

I must be a terrific teacher, because I've got literally hundreds of success stories in my files. I've spent three hours digging in the mess, but I can't seem to find those five, so I'll just grab five at random.

Frank: "I remember you told us that 'growth produces bigness,' but I never appreciated what that meant. After being promoted for our team's successes the third time in less than three years, I had lost my instinct for what was going on, and I couldn't figure out why.

"Eventually I realized that our Agile team was so close-knit we'd been cut off from all our former sources of information—in my case, leisurely lunches with the people depending on me—because I couldn't be allowed to have unprogrammed time.

"So I began to program in some unprogrammed time. It's not the same as in the old days, but it's better than it was a few months ago. I'm trying to convince my other team members to follow my example, but so far they just think I'm slacking off from our real work."

Iris: "We were all promoted because of my role in our team's work in creating and installing a system to operate our machine tools. Nobody had ever done anything like that before in our company, let alone a woman. I was truly proud of my achievement, but technology in the machine tool business moves fast. Within a year there were new developments that could have been improved in the system.

"When some of the people in my team suggested replacing part of 'my' system, I came down pretty heavily on them. At the time, I thought I was being perfectly rational. It was only much later when I saw through the smokescreen my own pride had created. In one year, I had gone from being the solution to being the problem."

Barry: "For the first seven years of my career, I moved from one crummy organization to another. Finally, I arrived here and found a place on an Agile team that really valued my technical competence. I was the primary lead on six major design efforts, which you'll understand kept me busier than I'd ever been before. In those other places, they didn't give me anything terrifically challenging, so I always had time to read, take courses, and go to technical meetings.

"Also, moving from one job to another at least broadened my vision, but now I was stuck in one place—with one standard way of doing things. It took less than two years to become a piece of technical fluff—a real lightweight."

Ahmed: "Over a period of two years on our Agile team, I rose from an unwashed college kid to one of the kings of systems programmers. I had participated in designing and creating such a toolkit of programs and techniques I could make that system sing 'Waltzing Matilda'—and dance to the tune. Then the company bought a new system. Our team was asked to tend the old one during a one-year period of parallel operation—one year that stretched to two. Obviously, we were the best qualified to do it, but we weren't excited about the lack of challenge. To motivate us, they made us a very attractive financial offer. I went along with it, at least partly because I didn't really feel like giving up my top position and starting over at the bottom with the new system.

"The problem came when the old system finally went to the scrap yard. I had no place to go but the bottom of the new system, and everybody except our team had a two-year headstart on us. I decided to take a non-technical position in operations. I don't particularly like it, though the pay is good."

Sue Anne: "For the first three years I was analyst/designer for the marketing department, nobody paid much attention to me. We converted the development group to Agile, and I became the customer surrogate for four Agile teams. Then all four of the systems I had worked on came online within a period of two months. They were all great successes, and suddenly everybody thought little Sue Anne was a genius—including little Sue Anne. I was given three new projects in quick succession, each representing a different department.

"When I was back in marketing, I had taken the time to get to know everybody really well on a personal basis. Now that I was so much in demand, my style changed. I was cutting people short and ramrodding my own ideas through meetings. I suppose I don't have to tell you the new projects went nowhere. One manager actually asked the development manager to have me removed. That really hurt, but it's what woke me up to what I had become."

Gee, maybe they weren't such great success stories after all. But these five must be exceptional cases. The Agile approach is so terrific that the majority of technical leaders working with Agile teams must surely glide smoothly from one triumph to the next, not letting themselves be weighed down by past successes. I'm sure I could prove that with cases from my extensive files of success stories, but I'm so busy keeping my published books up to date, I don't have time to keep them organized.

Success Can Breed Failure

Of course, I'm kidding. About my lack of organization, but not about this stories of Agile success—though the names and circumstances have been altered to protect the innocent. And Frank, Iris, Barry, Ahmed, and Sue Ann were innocent.

Innocent of what? Innocent of the way the world works on winners. You see, winning doesn't happen automatically. Boulding's Backward Basis says, "Things are the way they are because they got that way." [see, *The Secrets of Consulting*] None of these five winners really thought about why they had become winners until after they became losers. Too late, they realized that their competence had come from their experiences and environment, so that when their experiences and environment changed, so did their competence.

The Agile Manifesto Principles says several things about maintaining tech competence, starting with this:

• *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*

After becoming winners, our five heroes each worked at a non-sustainable pace, largely due to the Pile-On Dynamic. [see, *Quality Software 5, Managing Yourself and Others*] In this dynamic, the best-informed people tend to become overloaded. When they're overloaded, they tend to ignore any task that doesn't contribute instantly to the primary task.

In other words, they can no longer pay continuous attention to what made them best-informed in the first place, thus violating two more Agile Principles:

• *Continuous attention to technical excellence and good design enhances agility.*

• *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Success doesn't happen automatically. It's not Zeus granting you favorable treatment. It's not because you're just better than other people, so don't be smug. It's not just good luck, either. It's constant attention to what it takes to become more effective, but also adjusting your behavior accordingly. You succeed by paying attention, which our heroes did—until they didn't. You succeed by adjusting your behavior—but in accordance with what make you more effective. Our heroes did adjust their behavior—because their environment changed but they weren't adjusting their behavior.

When Agile was first introduced, many people though that issuing a Manifesto and some Principles was all that was needed to transform the way their organization developed software. As the years passed, more and more people realized that introducing Agile required more than saying the right words. But once they succeeded with the introducing, some of them forgot that agility must be maintained—and maintained using Agile principles. There can be no resting on your Agile laurels.

Back To Index

Biography

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

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

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

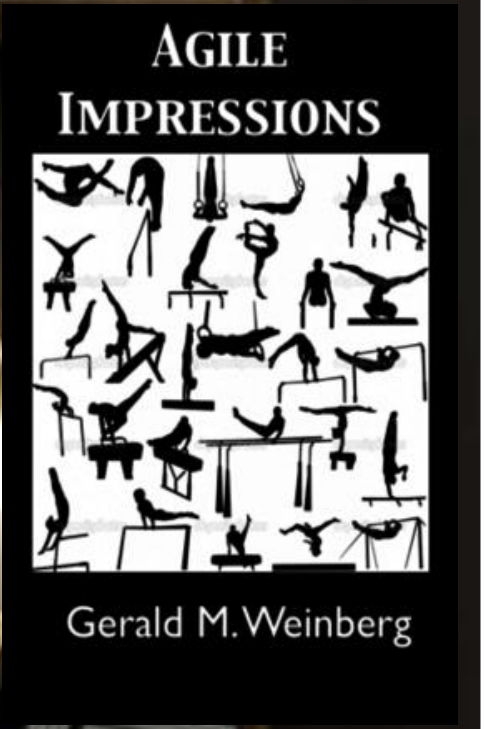
To know more about Gerald and his work, please visit his Official Website here .

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.

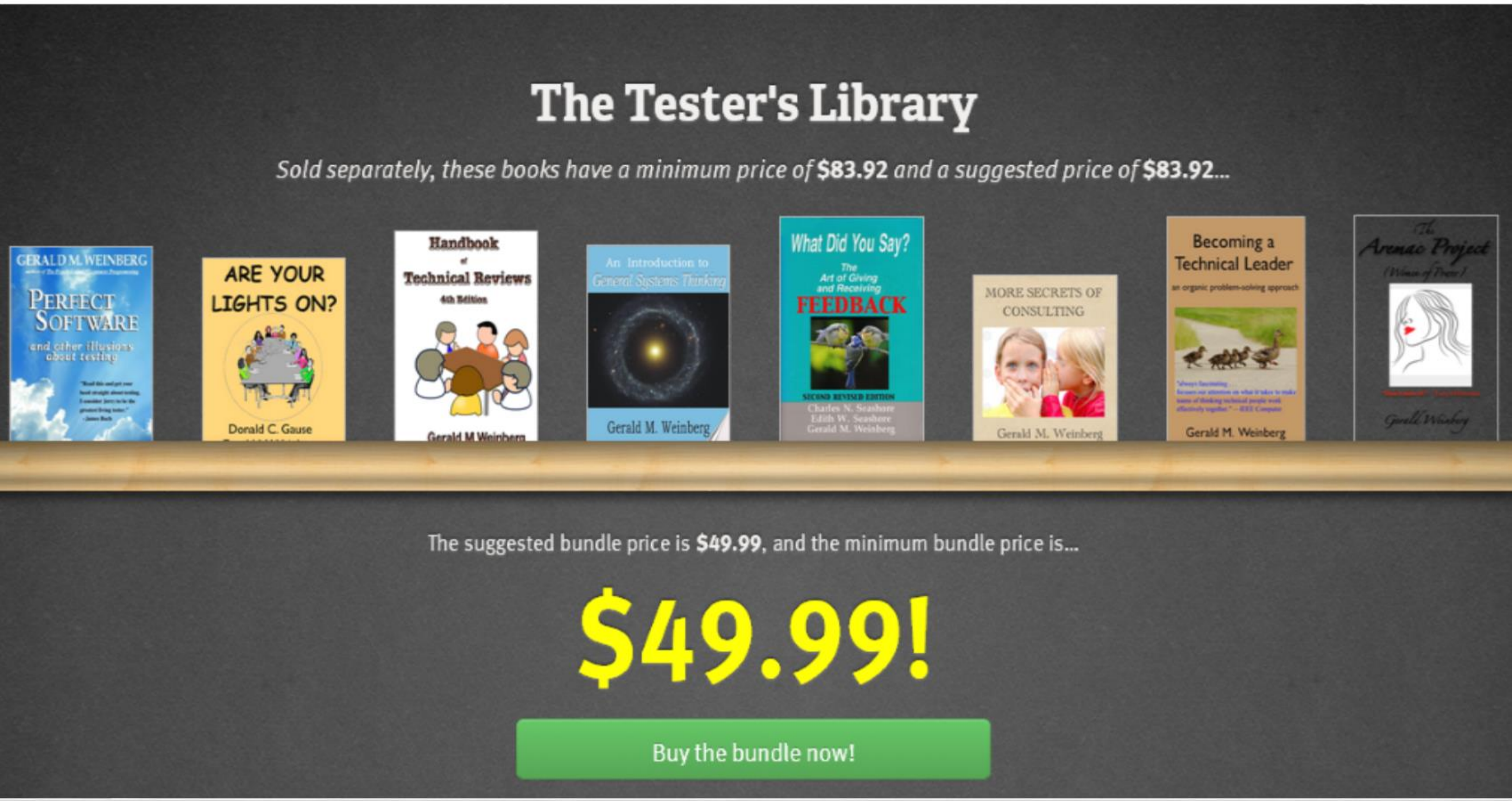
Know more about Jerry's writing on software on his website.



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!

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

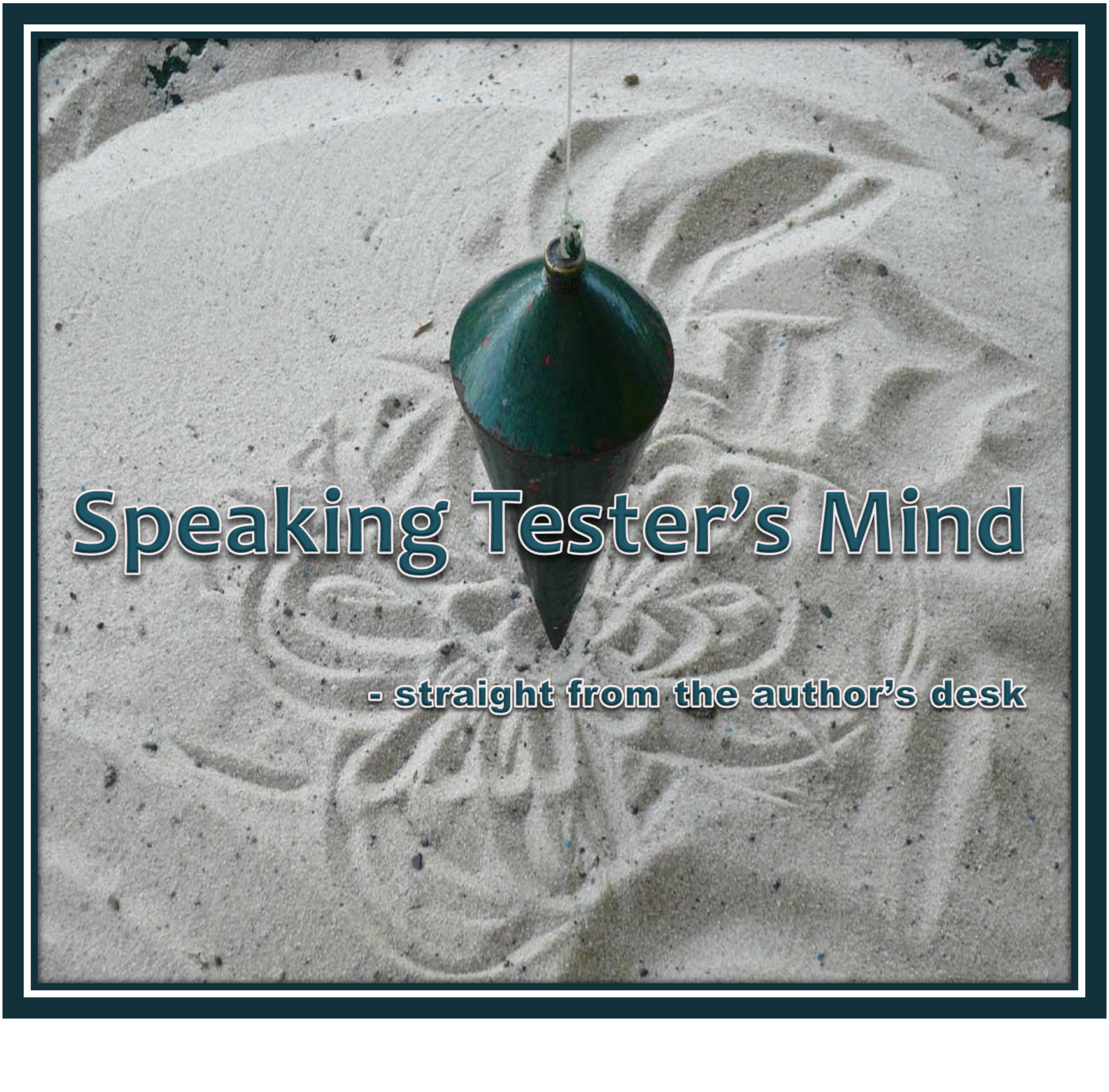
- What Did You Say? The Art of Giving and Receiving Feedback

- More Secrets of Consulting

-Becoming a Technical Leader

- The Aremac Project

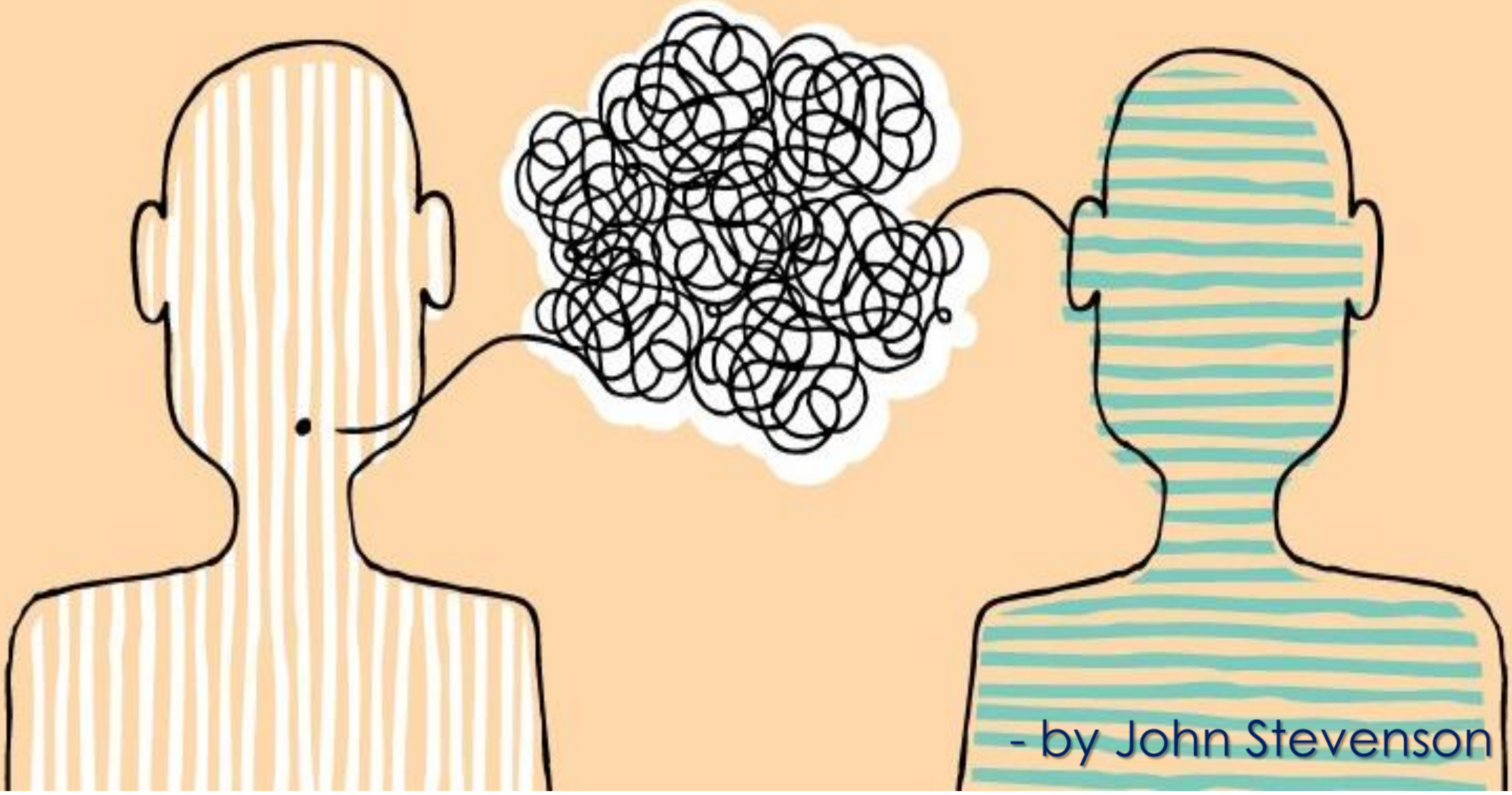
Know more about this bundle



Speaking Tester's Mind

- straight from the author's desk

Testing Skills
Part 2: Influencing by Listening



- by John Stevenson

This is the second in a series of hopefully short articles which looks at skills and techniques outside the traditional testing that can be useful to those who practice testing.

Future topics planned include:

- Note Taking
- Leading teams
- Persuasion and how to sell
- Speaking the language of business
- Remote teaching experiential style
- Going beyond the model

If you can think of any others that would be useful please let me know.

I am making these public, since by writing it down and making it public I am committing myself to do it. That is your first tip in this article, if you want to commit to doing something which you keep putting off, writing it down and make it public.

Having recently attended an excellent internal workshop on the' artistry of influence'. One of the key takeaways I got from the course was to be able to influence you need to **STOP** and **LISTEN** to what the person is saying and most importantly how they are saying it.

During the course I discovered that when communicating we have a preference for using one of three types of senses, sound, sight and touch. Which when converted into communication terms become auditory (sound), kinesthetics (touch) and visual (sight).

Do they use visual words such as 'see, bright, focus', kinesthetic 'feel, handle, hard' or Auditory 'hear, ring a bell, sound right'.

Once you uncover their prefer sense for communicating you can try and match their preferred sense. This is called pacing and its roots can be found in our tendency to form tribes and groups based around similar traits. This enabled us to survive as a group, since we shared similar habits and vocabulary. It also encouraged strong relationships within the groups.

It is also useful to establish the tonality of the person you are listening to. Are they talking quickly, loudly, slowly or quietly? Ask a friend to help you practice by matching the voice tonality and their preferred sense. By listening to others and the way they talk you can start to establish a rapport and build a relationship in which they can be influenced on a subconscious level.

Be careful when influencing others that it does not become more than just trying to persuade others of your thoughts and ideas. If people feel or find out they are being manipulated then they will probably loose respect for you and you'll destroy any possibility of a working relationship. The key aspect when trying to influence people is to ensure that they know it will have a positive benefit for them, even if at the same time you will gain a positive benefit. The person you are trying to influence has to see a benefit in it for them.

It is not about **YOU** it is about **THEM**.

There is a lot more involved in using listening as an influencing tool and this is only an introduction.

Let me know how you get on with this tip on influencing by listening.

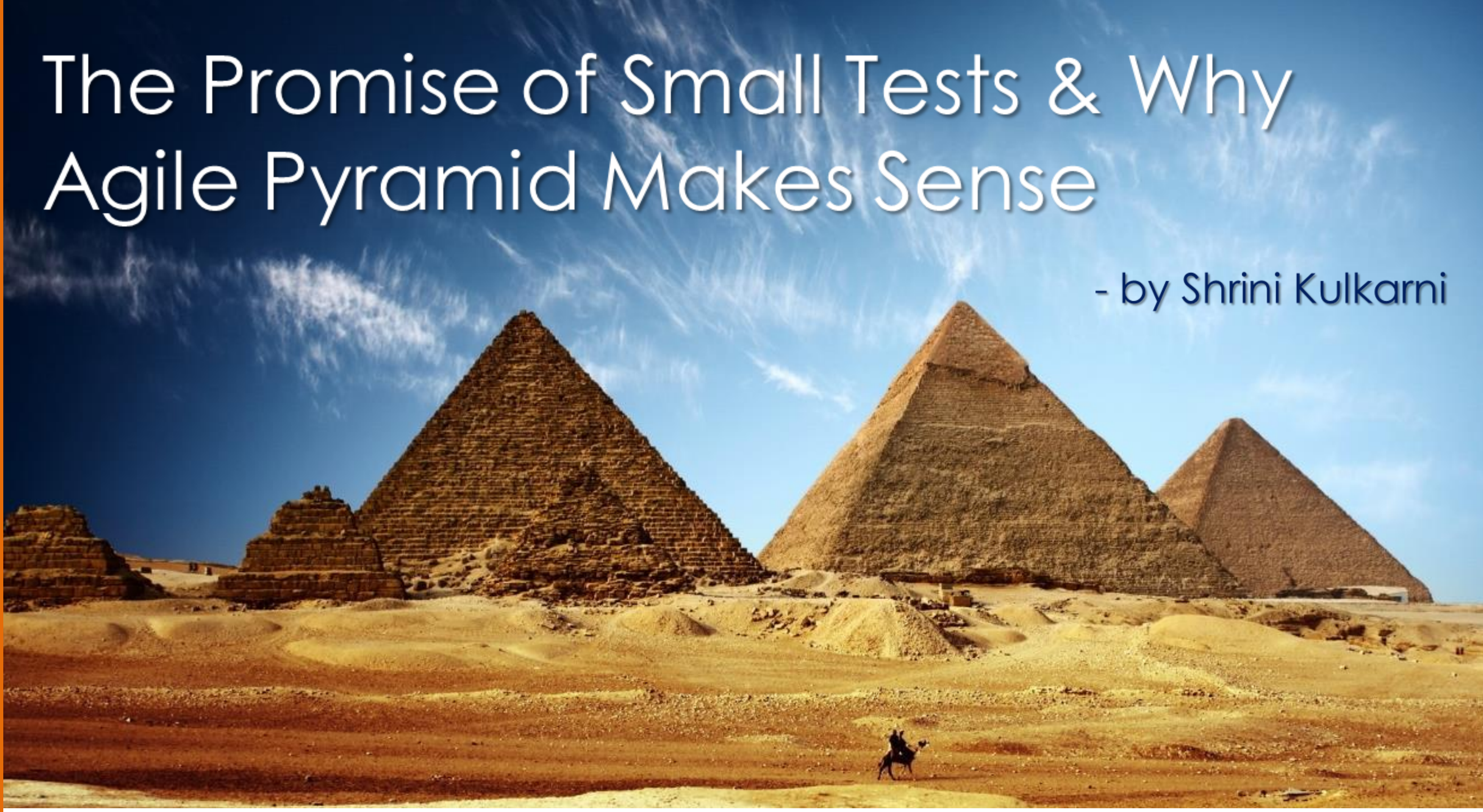
Further reading:

- Real world applications of pacing and leading
- How to persuade anyone using pacing and leading
- What is NLP
- Influencing skills: a how-to guide, or, How to get what you want without making enemies

Back To Index

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 Promise of Small Tests & Why Agile Pyramid Makes Sense

- by Shrini Kulkarni

Introduction

This article is about small tests aka unit tests and the promise that they hold. You might have heard this piece of advice that if you are writing code you must write tests – unit tests. In the overall automation landscape, small tests have a special place and act as foundation for all automated tests of other category. The article also dwells on agile pyramid and reasons for the recommendations it makes.

Manual Test v/s Automated Test

For a moment, let us take a completely different and basis view of "test". A test could be specified in a language that is meant for human tester – written in human language or presented as a diagram, data table or say mind map. Such type of test is typically referred as "manual test". When a test is specified in a programming language meant for a computer – it is referred as an automated test. Automated tests could be written targeting different levels of focus and accordingly classified as small, medium sized and large tests. While small tests are your good old unit tests that lock down all dependencies, mock external interfaces and focus only validating logic at smallest level of code – say a method, medium sized tests relax the lock down of dependencies and use the actual interfaces to mimic integrated system at say, API level. Large tests validate full system along with all interfaces and upstream/downstream systems connected to system under test. As size of test increase, tests tend to run slower and become difficult to troubleshoot failures. Since all automated tests use algorithmic way of deciding the result of the test without needing any cognitive abilities of a skilled human tester, as Michael Bolton suggests, are "automated checks".

Why tests (automated) fail?

When you run a bunch of automated tests, you get some results. If you can trust the results – especially the **pass** ones and you are sure that failed ones indicate some problem in the system under test - then this automation adds value to overall testing. I have often seen people ignoring the amount time one spends to analyze and reconcile the results after automated tests are run. A poorly constructed automation suite gives results that are unreliable and unrepeatable and as such automation is an overhead instead of a help to testing.

First of all a failed test does not (always) mean a problem in the system under test (SUT). The real purpose of having tests is to programmatically validate or assert some expected behavior. In that sense, a failed test is failed validation of expected behavior.

A "true" failed test in one sense indicates a defect in the system. In one sense a failed test has succeeded in its objective of spotting a problem. A "true' passed test gives/enhances the confidence in the system that specific functionality as validated by the test is "working". One of the main goals of good automation is to ensure that tests are reliable – both passed and failed ones can be trusted.

Let us look at why tests fail - there could be multiple numbers of reasons for failed tests ranging from trivial to conditions that test cannot control.

- Test data set up related issues – this is biggest and most tricky reason to control. A test may fail as SUT is configured to bring about a behavior that is being validated.
- Prerequisites and dependencies – failures due to supporting systems or interfaces not configured as per the need of test in question.
- Intended and unintended changes in AUT – in GUI automation this condition manifests as screen object changes, id/label changes of objects in the GUI. There might be a new change in the AUT behavior that has not yet been implemented in test (maintenance issue). In rare occasions, failed tests also indicated an "unintended" code change that occurred as consequence of "intended" change elsewhere in AUT.
- Test environment issues – network speed, memory/disk space on server, browser or device settings. The test environment related issues cause intermittent failures in the tests that do not point a problem in SUT

How to increase the reliability of tests (results)?

The small tests are best examples of highly reliable tests. They attempt to remove or mock all dependencies that a test may have and focus solely on logic of an isolated piece of code. Good small tests need to adhere to following rules/constraints to be called as small (unit) tests

- Each test is isolated and independent
- Each tests creates through mocking all test data and collaborators so that there is no failure due to data set up
- Tests do not require specific deployment or configuration of SUT – they can be run straight out of dev machine/environment
- Tests do not access any resource on local hard disk or across the network

As you can see – through these constraints, every possible attempt is made to make sure that a failure will be indeed a problem or potential problem in the unit/piece of code of SUT being tested. Due to high reliability of results, developers can trust passed tests that indicate nothing (more or less) has changed in the code (which may or may not lead to a bug/defect).

Small tests as Change (unintended) Detectors

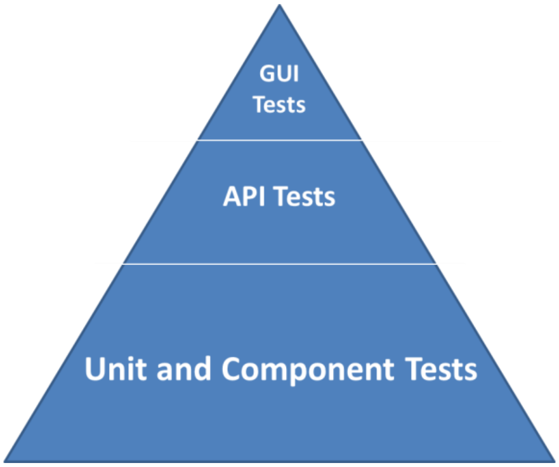
Unlike other types of tests that aim at identifying bugs/potential problems, small tests are designed to capture unintended code changes. A failed small test would surely flag off something has gone wrong logic of piece of code. Making use of faster feedback cycle – developer can quickly investigate a failed test to see if something changed unintentionally. Thus instead of looking for bugs, small tests look for changes in the code that may or might not be bugs/problems yet. Passing small tests would confirm that a smallest level of code, there are no unwarranted changes. In order to perform this role of "small change detectors" - small tests needed cover /all/ possible paths within the code. We need to maintain a very high level of code coverage in order to live up to the promise of high reliability of these tests.

Small tests = speed

While one might shrug off the complaint of slow running tests at any level (small, medium and large tests), one of the key advantages of small tests is their speed – several thousands of them can be run in matter of few seconds. Small tests can be run straight from developers IDE without requiring any set up or configuration. They can be used as last "quality" check before code is checked into Source control system. This ensures quickest possible feedback to developers and more stable code trunk. Developers can determine what got changed as even before committing the code to source control system. The speed coupled with high degree of reliability – makes automation work very efficiently and effectively.

Why Agile Pyramid advocates what it does

Importance of small tests and the role they play in overall landscape of automated tests is nicely represented in Agile pyramid (or triangle?). Agile Pyramid is an idea first introduced by Mike Cohen and it provides guidance on how your tests are distributed across three buckets – small, medium and large tests. Given high degree of reliability and speed of execution – no wonder, they should constitute a large portion of your overall tests. While it might make no sense in putting a number or percentage on what portion of small tests makes a recommended portion (counting test cases is like counting unicorns – totally waste of time), One way, I would read the advice as large number of small tests in relative terms to other types of tests.

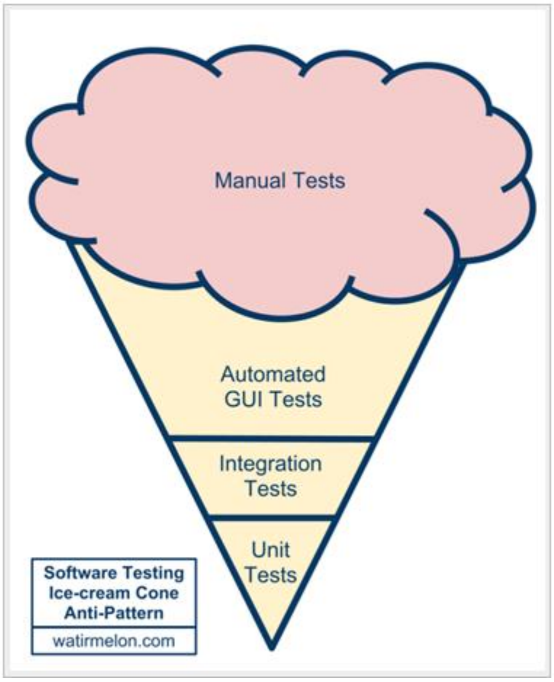


What happens as we traverse upwards to the vertex of pyramid - we would start deviating from constraints of small tests and start accessing resources across network, start relying on certain data setup, we would start accessing third party applications and interfaces, etc., What would that result into? Reduced reliability of results of the tests - as now tests could fail on reasons other than a potential bug/problem in the SUT. Medium and large tests validate integration between smaller components validated by small tests. These tests would run slower and many of these need code deployment.

We need medium and large tests - we need to be sure about their role and limitations.

What about GUI tests then? Should we stop developing them?

For over decade or two – Automation was mostly meant as GUI automation. The popularity of GUI automation mainly amongst executive management made sure the market for them was sustained and flourished. GUI automation was easily demonstrable through record-playback approach taken by many commercial automation tools. But surprisingly the whole baggage of problems these tests brought into is systematically ignored.



GUI tests are probably the reason why ice cream code anti pattern was discovered to describe a condition of having tests represented in inverted agile pyramid. Having large number GUI/large tests and very small number small tests is regularly observed practice in waterfall projects.

While writing about GUI automation and its problems would require a separate post/article, the benefits and challenges can be easily worked by comparing them to small tests. First of all, amongst the tests the results from GUI tests are least reliable – thanks to their heavy dependency on data setup and need for programmatically identifying GUI objects on the interface.

GUI tests need GUI interface to be stable and consistent in order for the tests to produce reliable results. In this way they push automation late in the cycle. GUI tests run slow and mostly need code deployment on test environments. The staging required to run and obtain reliable results for GUI tests is so time consuming and laborious that performing skilled human exploratory testing will be relatively effective and fast. Of course you need intelligent human tests with freedom to get best out. Please keep in mind that not all GUI tests unless otherwise designed – are not large tests. Most of GUI tests begin and end in the GUI interface only and validations done at GUI (only) are often not reliable.

Summary

I wondered why Agile pyramid makes the recommendations it makes. I asked why we need to have large number of small tests as compared to GUI (large tests). The trail of thinking and analysis that I followed is presented here. Speed and reliability are two main considerations that make small tests as “must-have”. Large tests (GUI tests) while are expected to do the job of exercising full application mimicking how an end user would use the system, tend to be flaky, slow to run and often very difficult to troubleshoot. Thus for an effective automation, it would be wise to follow recommendation of Agile pyramid having large number small tests and small number of large tests.

Shrini Kulkarni is Test Evangelist (over 18+ Years of experience) specializing in Enterprise level software Testing, Test Automation, Mobile Automation, Agile Adoption and Performance Testing.

As experienced Tester, Test Manager and Testing coach/consultant Shrini can help organizations in assessing their testing, automation and performance testing practices and develop skills.

As a popular blogger on Software Testing, Shrini's writing has attracted many software testing professionals around the world.

Shrini has worked in companies like HCL Technologies, iGATE Global Solutions, Microsoft Hyderabad, Aditi Technologies Bangalore, i2 technologies Bangalore.

Shrini worked in many roles covering entire spectrum of IT – like Software developer, Project lead, SQA lead, Test Lead, Test Manager.

Follow Shrini on Twitter @shrinik or connect with him on LinkedIn - https://www.linkedin.com/in/shrinik

Back To Index

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

IT – Compliance for Test Data Management

- by Rozalia Noga

IT - Compliance Manager

"A journey of a thousand miles begins with a single step!"

Lao-tzu

Who is the IT - Compliance Manager?

The **IT - Compliance Manager** is the interface between the operating departments, the IT software development and the quality assurance. The first objective of IT - Compliance Manager is the regulatory compliance and corporate policies. He is responsible for ensuring that there are no legal implications concerning the company and always to acts in the interest of the company.

The IT - Compliance Manager has **interdisciplinary knowledge** based on the law referring to IT. These include COBIT, IT Security (IT Security Management), current law etc. and links the two disciplines together.

Furthermore, the IT - Compliance Manager represents an example for the company, departments and employees. The article "implementation of the compliance program within the company and in the business processes from Hubertus Eichler / Dr. Ulrich Vogel / Patricia Krautner (www.ZRFCdigital.de ZRFC 1.13) describes and points out, which characteristics these role models should possess for this position and which challenge he has to face.

Compliance - Manager
Stable system of values and high principles orientation
Sincerity
Good communication skills
Motivation and enthusiasm
Care and diligence
Responsibility
Assertiveness
Good people skills
Knowledge of and dealing with complex contexts
Internal balance
Good reasoning ability
Consistent action, even against resistance
Independence

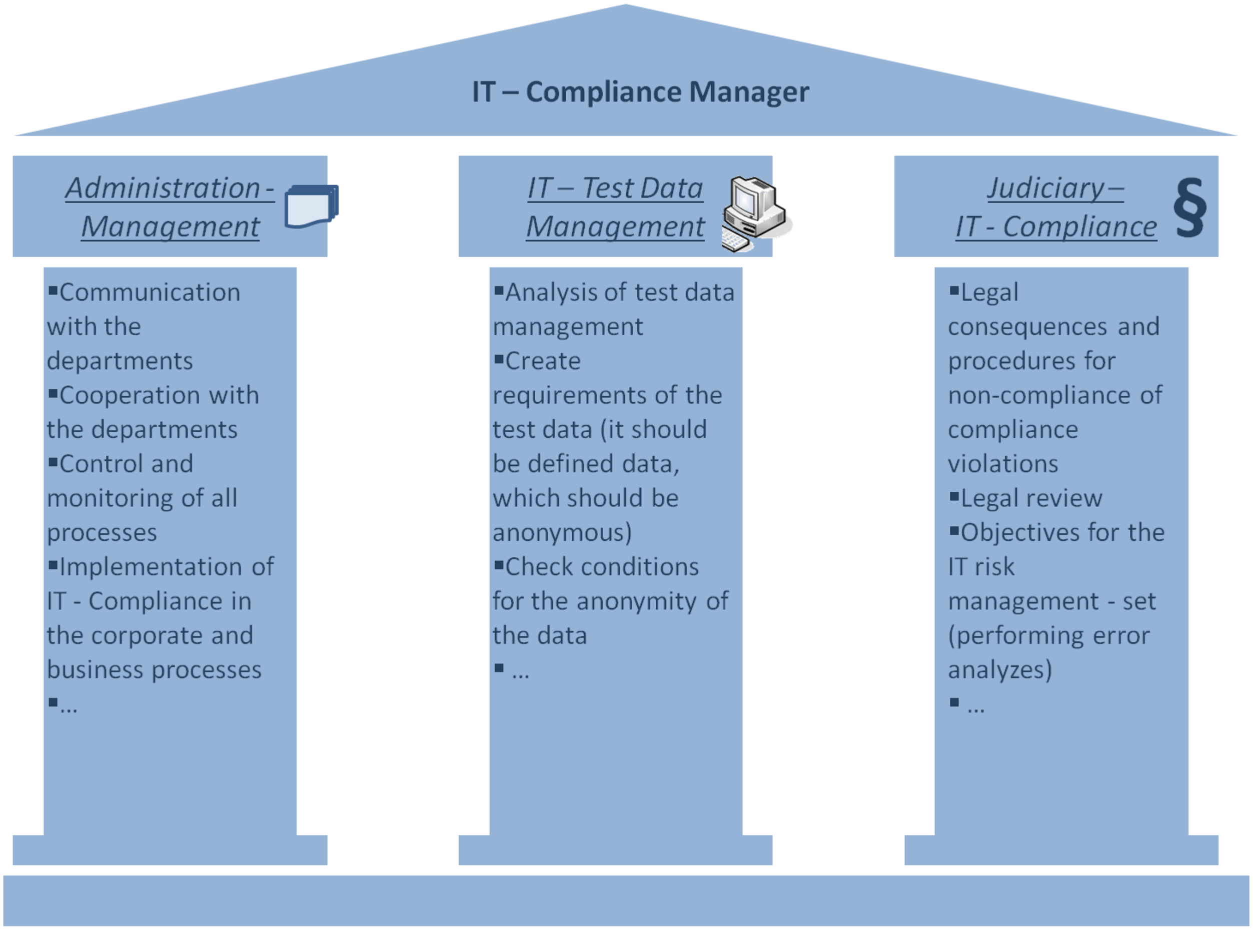
Quelle: www.ZRFCdigital.de ZRFC 01.13

The characteristics listed above show that the position comes with a great responsibility and challenge. Therefore, this position is suitable for staff members who have years of professional experience in both disciplines.

What are the responsibilities of the IT – Compliance Manager?

The IT - Compliance Manager covers an extensive area of responsibility, being an interface between the departments and the IT (software development and quality assurance). One of the main tasks of the IT - Compliance Manager is to **"transfer the IT - Compliance to the natural subject of everyday business processes".**

There are still many other tasks which are broadly divided into three sections: Management, IT and Judiciary.



IT – Compliance Manager
Administration - Management
IT – Test Data Management
Judiciary – IT - Compliance
§

- Communication with the departments
- Cooperation with the departments
- Control and monitoring of all processes
- Implementation of IT - Compliance in the corporate and business processes
- …

- Analysis of test data management
- Create requirements of the test data (it should be defined data, which should be anonymous)
- Check conditions for the anonymity of the data
- …

- Legal consequences and procedures for non-compliance of compliance violations
- Legal review
- Objectives for the IT risk management - set (performing error analyzes)
- …

1. Administration - Management

- Communication with the departments
- Cooperation with the departments
- Control and monitoring of all processes
- Implementation of IT - Compliance in the corporate and business processes
- Creation of IT - Compliance regulations
- Compliance - create requirements
- IT - Compliance – establish culture in the company
- Training of staff with the handling of data, staff awareness of data (behavior with data, fraud, breach of internal regulations, disclosure obligations)

2. IT – Test Data Management

- Analysis of test data management
- Create requirements of the test data (it should be defined which data should be anonymous)
- Check conditions for the anonymity of the data
- Developed anonymization method (the time of anonymization = regular process)

- Check availability of test data and test data volume (the validity of the test data must be guaranteed)
- Perform regular audits
- Provide periodic data quality analysis
- Vulnerability analysis based on the test data
- IT – Security Management, IT - Security Standards
- Storage of the original data
- Digital data management (legal requirements)

3. Judiciary – IT - Compliance

- Legal consequences and procedures for non-compliance of compliance violations
- Legal review
- Objectives for the IT risk management - set (performing error analyses)
- Observe safety policy - aspects of IT
- Improvement and development of IT - Compliance (if applicable design best practices, checklists)
- Data protection commissioner - legal data protection requirements
- Liability of corporate governance in violation of IT - Compliance requirements

Nowadays, companies want all relevant safety requirements to be met without any legal conflicts, therefore, a responsible employee or IT - Compliance Manager is required to control the legal aspects and to protect the company from damages.

The tasks explained in the three columns are important components and affect the future software and also the company's success. The IT must meet the constantly growing demands. The IT - Compliance Manager is responsible for ensuring that the data is protected and the (internal and external) guidelines are followed precisely. To ensure the quality of the test data and the tests, the implementation of IT - Compliance Manager in the test data management is another important aspect. The IT - Compliance Manager can look after test data management in its entirety. He should be the first person to be questioned concerning all aspects of test data management and IT- safety.

The message of the IT - Compliance Manager in a company is to indicate the staff’s attention and to exemplify IT - Compliance. For this reason, he represents a model for the entire company.

Challenges

"There´s a way to do it better!"

Thomas A. Edison

The major challenge is the communication process. The test team needs testing data and they have to communicate with various people to explain why they need it. Sometimes the data that the test team gets, is far from being useful. The process needs more time than test team can wait for. It is very difficult to find the balance. One way to overcome this challenge is to implement the IT – Compliance Manager in the communication process for the test data management.

Example: How to overcome the challenge

The role of IT - Compliance Manager is to maintain control and the entire overview of the test data management. He should be the first person to be questioned concerning all aspects of test data management and IT- safety. -For this reason, the implementation of IT - Compliance Manager between the departments and the IT Department (software development, quality assurance and test data management) is made.

Communication process

The quality assurance or even the software development reports to the IT - Compliance Manager, that data is needed for testing. He is notified about tests and data that is needed. Thereupon, the IT - Compliance Manager contacts the departments. The data requested can be from various departments, such as accountancy, sales and distribution. Although this depends on the size of the company and which operational departments exist in a company.

The IT - Compliance Manager clarifies together with the departments which data is needed and when. He checks all the required conditions, so that the data can be used for testing purposes, containing a high expressiveness and the quality of the data is ensured. The anonymization of data takes place and the data is transferred to the quality assurance and also to the software development. The tests can thus be performed.

Conclusion

> *“First they ignore you, then they laugh at you,*
>
> *then they fight you, then you win!”*
>
> ***Mahatma Gandhi***

The test data management is becoming increasingly important, and will be even more important in the future. The companies disregard this fact and treat the test data management with less importance. The goal should be that every company should have an IT - Compliance Manager as a standard. He is responsible for realistically assessing the risks incurred by the company and implementing the necessary processes as well as control systems for the test data.

IT - Compliance requires time and money but its absence can also cost time and money. In order to gain a competitive advantage in the future and to bring high quality software products in the market, the awareness should be raised and investment on it should be made before it’s too late.

References

[1] DICO – Deutsches Institut für Compliance http://dico-ev.de/index.php?id=86

[2] NIFIS – Nationale Initiative für Informations- und Internet-Sicherheit http://www.nifis.de

[3] Compliance Magazin www.compliancemagazin.de

[4] TeleTrusT – Bundesverband IT-Sicherheit e.V. https://www.teletrust.de/

[5] Wikipedia http://de.wikipedia.org/wiki/IT-Compliance

[6] ZRFCdigital – Risk, Fraud & Compliance www.ZRFCdigital.de

Implementierung des Compliance Programms im Unternehmen und in die Geschäftsprozesse von Hubertus Eichler / Dr. Ulrich Vogel / Patricia Krautner (www.ZRFCdigital.de ZRFC 01.13)

[7] SearchSecurity http://www.searchsecurity.de/

[8] http://www.compas-ict.eu/project.php http://www.compas-ict.eu

[9] Testing Experience www.testingexperience.com testingexperience16_12_11.pdf testingexperience17_03_12.pdf

testingexperience22_06_13.pdf

[10] Book: Kurt Schneider

"Abenteuer Softwarequalität: Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement"

[11] Book: Ralf-T. Grünendahl, Andreas F. Steinbacher, Peter H.L. Will

"Das IT-Gesetz: Compliance in der IT-Sicherheit: Leitfaden für ein Regelwerk zur IT-Sicherheit im Unternehmen"

Rozalia Noga works as a Software Test Engineer in scrum environment. As a student she worked for SAP Germany and started there her career as a Software Test Engineer.

She has testing experience in industry and commerce software, medical software and mobile app testing.

Rozalia is a Certified Tester of the ISTQB Foundation Level (CTFL).

Back To Index

Do You See a Problem?

- by Albert Gareev

Not That Apple

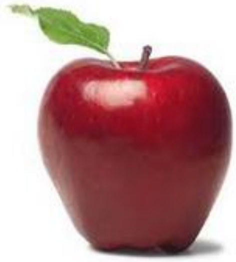
Do you see a problem? With the image?

Maybe you don’t. You know, people who can’t see still can use software, browse the Web, and work with electronic documents. Screen Reader[1] is a technology that makes all these possible.

But a screen reader can’t “see” this image of an apple, either. It would be able to say a text alternative[2] if it was provided. But for this one I’m not putting ***alt = “apple”*** to illustrate the point.

Here’s another image and let’s say now I do provide the text alternative.

[1] Screen Readers: https://en.wikipedia.org/wiki/Screen_reader
[2] Text alternatives: http://www.w3.org/TR/UNDERSTANDING-WCAG20/text-equiv-all.html



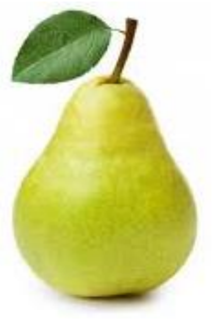
Do you see a problem?

Oh… That’s another apple? Another image of another apple, to be more accurate?

But how did you know? You saw?

But screen readers can’t see. Neither can checking tools. So called “accessibility testing tools” that scan HTML of the page and check for pre-programmed patterns[3] most likely will report an image without ***alt*** attribute, but as long as it has some text it’s a “pass”.

Let’s try again. Here’s another apple.

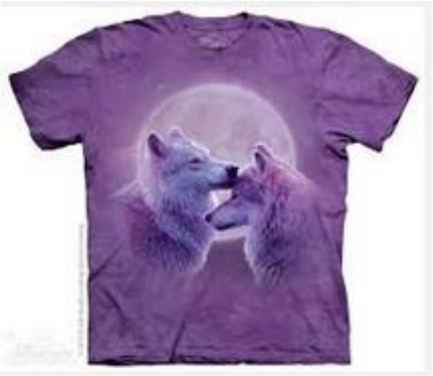


Do you see a problem? Okay, it’s not really the image of an apple.

How to report these? I suppose, we can describe that the image of a pear has invalid text alternative. But how to describe a difference between the first and the second apples? What would be a valid text alternative? We may describe that one has leaf turned right and another – turned left. “A red ripe apple with green leaf turned right”, maybe?

Keep in mind we’ve dealt with relatively simple objects. Now think of a web site of a clothing store. Shirts and shirts.

[3] All HTML checking tools are application agnostic; they check HTML syntax but not specific values



How can you describe THAT?

Let's try. "Shirt with image printed – moon, 2 wolves".

I didn't mention it's a full moon. I didn't mention that it's partial images of wolves. I didn't mention about what they do. Nor that I could. I can guess they're not fighting. Anything more specific would be subjective and imaginative. It's just a picture, after all.

Now, let's say we came up with a 100 words long paragraph of textual description as alternative. Go back to our imagined clothing web site and try to navigate through a page showcasing 10 shirts. How long will it take to listen to every detailed description?

Apparently, too detailed text poses a problem, too…

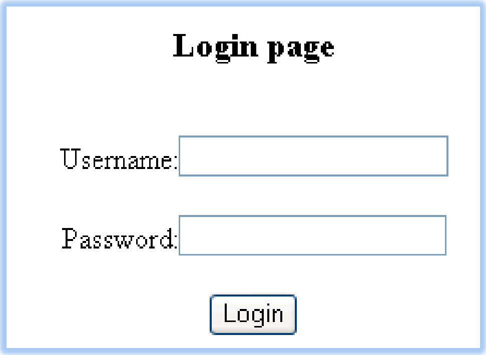
So, checking whether an image has an alternative text or not, is easy. WCAG[4] has a straight forward definition:

- *Non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose.*

Or is it?

What's In a Label?

Do you see a problem with the login form represented by the image?



[4] WCAG – Web Content Accessibility Guidelines: http://www.w3.org/TR/WCAG20/

There are a few, actually, but we can't know for sure because it's just an image; we have no HTML to check.

That's another testing problem by the way – accessibility testing of any prototypes like "UI mock-ups", "wireframes", "and sketches" is quite limited because we won't know if the implementing technology has accessibility features included.

But here's one specific problem I want to talk more about.

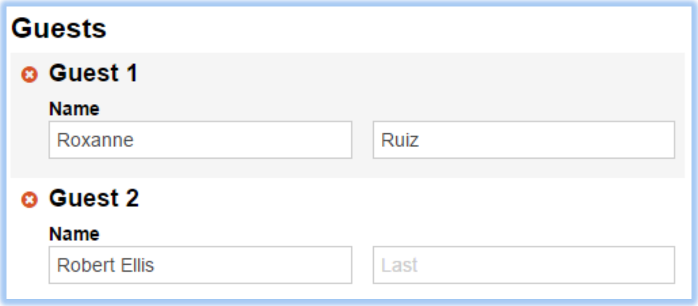
We have two requirements from WCAG.

- *2.4.6 Headings and Labels: Headings and labels describe topic or purpose.*
- *3.3.2 Labels or Instructions: Labels or instructions are provided when content requires user input.*

Sighted users could ***see*** both label and edit box. They may infer: because the two objects are visually connected it's most likely the "Username" edit box.

A screen reader won't see the visual connection and will never infer anything. Unless the HTML has specific attributes for linking the control and the label – ***"id"*** for the control, ***"for"*** for the label – the control is just an edit box with no description.

Now, let's take another example.



Guests

Guest 1

Name

Roxanne

Ruiz

Guest 2

Name

Robert Ellis

Last

Do you see a problem?

While sighted users may infer that "Name" label adjacent to the edit box means it's edit box for name, even they need to make another logical leap, guessing that the second edit box with no label at all might be connected to the first one. And together they probably stand for First Name and Family Name.

These assumptions do put some expectations on the users, as such below.

- Familiarity with UI patterns, i.e. certain abilities to memorize and recall them, as well as abilities to match the patterns
- Certain cognitive abilities to connect information and make conclusions

Looking again at the form now. If both edit boxes have label "Name" how to tell which one is which if you don't see them? How a non-sighted user can distinguish where they belong to? Same as in the previous example with apples, distinguishing between similar objects becomes a problem.

Furthermore, we have the same problems with checking tools.

- They may identify an ***input*** control without ***id*** attribute. Though the tool also has to match the ***type*** – text, button, checkbox or something else
- The tool may find a ***label*** tag with a corresponding value of ***for*** attribute
- Any label text would be a "pass" with the checking tool though
- The tool can't really evaluate the adjacency
- The tool can't identify if two labels mistakenly point at the same control, or had pointers mixed up (First Name instead of Last Name and vice versa)

Again, we cannot rely upon checking tools for good verification for the purpose. Best they can is to find an absence of some attributes. They can't even check if appropriate connection was defined.

P-A-S: Present - Appropriate - Sufficient

With the two chapters above we began with seemingly simple examples and seemingly clear requirements. As we started looking into details we uncovered clusters of problems.

It is clear now that the tools can only help with syntax checking. That sometimes they won't catch even otherwise obvious problems.

We are left with the old good skilled human way of testing.

How skilled testers recognize a problem? They apply *heuristic oracles* – fallible but effective and practical principles for decision-making[5].

How these heuristic oracles come to be? We *develop* them based on our experiences and analysis of patterns[6].

Let's try this exercise now – based on the examples reviewed.

1. In order to make non-text element of a Web page or a document accessible with Screen Reader a technological solution is required – special additions in mark-up language. It might be *alt* attribute for images, *label* tag for controls, and so on. Whatever sighted users *see and infer* based on the familiar UI patterns must be explicitly *present and coded* for screen readers to pick up the relationship between the document elements.
2. It does matter to use the right implementation techniques. For example, providing input instructions for an edit control in a form of *alt* attribute won't help because screen readers are hard-coded to support only specific patterns. Furthermore, all relationships between the elements must be *appropriately* designated – pointing labels to the right controls, providing valid *id* attributes, and so on.
3. Text alternatives must be *sufficient* in the context. They must describe the details important for the purpose and must not confuse with irrelevant details. For example, if it's online shopping with a choice of apples, providing means for distinguishing between green sour "Granny Smith" kind and red sweet "Gala" kind is important. But if it's an illustration to "apples and oranges" debate, one word description is sufficient. Whereas an image intended only to catch attention of a sighted user and having nothing relevant to the article should have nothing as the text alternative.

[5] Recommended reading – "All Oracles Are Heuristic", Michael Bolton: http://www.developsense.com/blog/2012/04/all-oracles-are-heuristic/

[6] Recommended reading – "Oracles from the Inside Out" , Michael Bolton: http://www.developsense.com/blog/2015/09/oracles-from-the-inside-out/

In short, remember these key criteria.

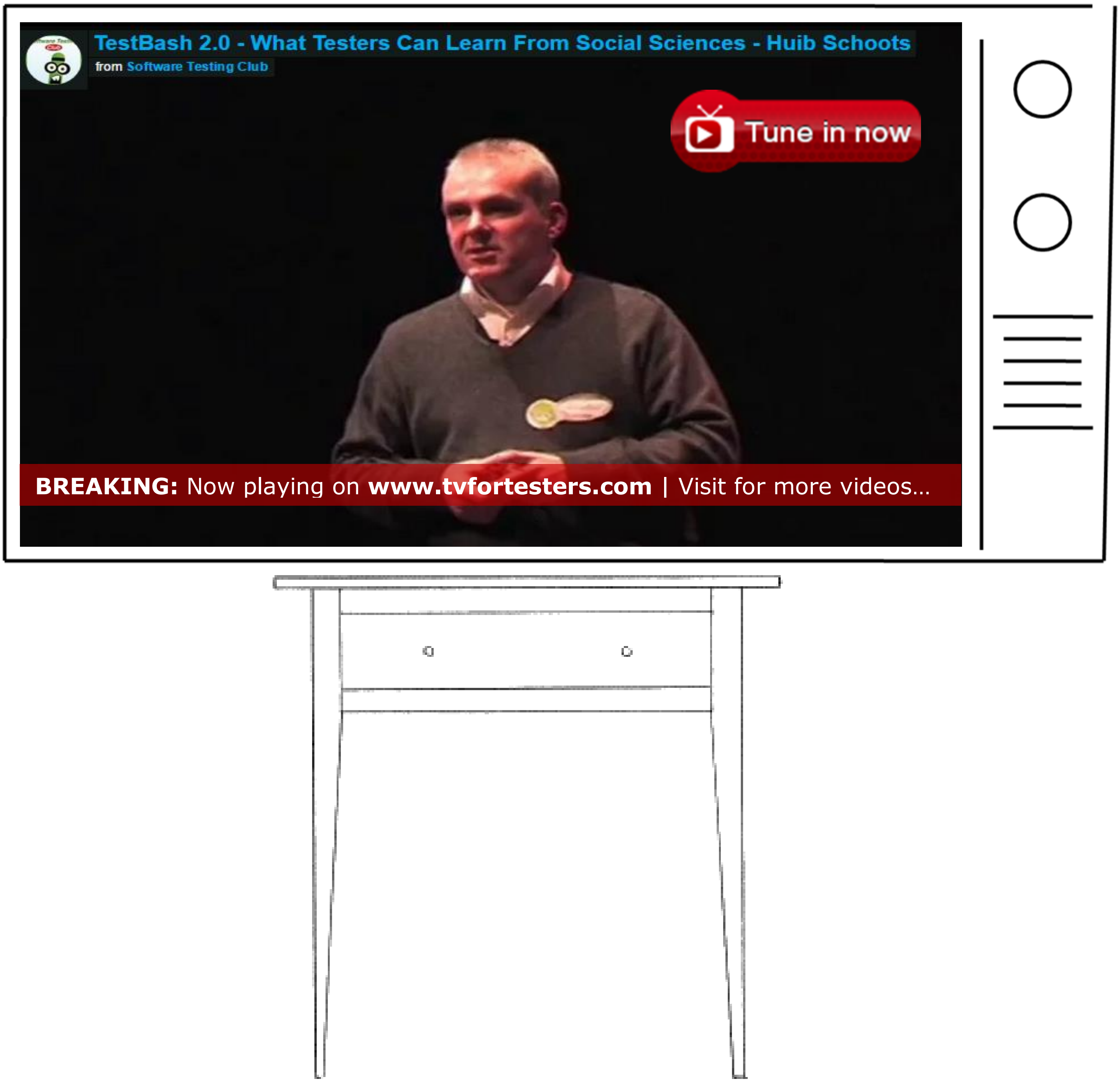
- ***Present*** – a tag or an attribute must be present in the document mark-up.
- ***Appropriate*** – it must be used appropriately in the technical sense.
- ***Sufficient*** – the textual alternative must be sufficient in the context.

Use "P-A-S" to "pass" the test or identify a problem, even if you don't see it with your eyes.

Albert Gareev is software testing craftsman and practice lead; Toronto based consultant and contractor. With over 20 years of diverse industry experience, Albert led testing for challenging projects; successfully implemented complex and large-scale automation. Albert is a "full stack" technical lead – he combines strategic and tactical vision; has exceptional hands-on skills; ensures continual growth of skills for the teams.

Albert is founder-organizer of Toronto Testing Meetup (http://www.meetup.com/Weekend-Testing-Toronto-Meetup/), and also a facilitator of the Americas chapter of Weekend Testing. Visit his blog at http://automation-beyond.com and follow @AGareev on Twitter.

Back To Index


TestBash 2.0 - What Testers Can Learn From Social Sciences - Huib Schoots
from Software Testing Club
Tune in now
BREAKING: Now playing on www.tvfortesters.com | Visit for more videos…

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

Give up to Grow

Recently I conducted a masterclass for a testing team as part of test design improvement initiative by the company. During the class I discovered that some of the tests that they do are also done by their developers. Well I am not sure if they were aware of these duplicated tests, but when I asked if they would stop doing this since their developers already did these, the answer surprised me. They said "Maybe not" as they were not sure if the developers did a good enough job. They did not trust their developers to do these tests effectively and therefore would prefer to ensure this, by doing it themselves.

And here it was, the first step to design improvement was not to do what was already done but do something different. In this case, the duplication was not a trivial number. Removing this duplication would put more time in the hands of these test folks, allowing them to do the new stuff that was being discussed in the class. So I told them "If you need to grow to do bigger/better things, you need to give up some".

The month of December in the Gregorian calendar is a wonderful month, it forces us to reflect what we have done during the year, and it makes us think of new goals/resolutions for the New Year and to look forward to an exciting new year of growth in our lives. To make way for growth, it is necessary to give up something.

Let us dwell on this phrase "Give up". What do we need to give up? It could be activities that we do, the current knowledge that we have, the mindset that we possess, the method that we use to learn and get better. Ultimately it is about transforming what we do, how we get things done, what we know, and how we acquire new knowledge and stay inspired.

GIVE UP

some ACTIVITIES that we do
—> to do NEW activities
—> to delegate and manage, not do
—> to inspire others to do

some KNOWLEDGE that we possess
—> purge to renew

the MINDSET that we have
—> change to become better/stay relevant

the areas that we LEARN
—> go outside your discipline, to seek inspirations from others

It is necessary to stop doing some activities by transferring these to others via delegation. Managing to ensure that transferred work is done well is key to success, a shoddy work would only worsen it. If you are delegating, setup a good system to ensure that outcomes do not suffer. Now that some work is off your hands, it is now possible to do new stuff and accomplish more. The ultimate is of course to inspire others so that they can come up with new things to be done rather than just take some load off you!

The knowledge that we possess needs to stay relevant with new times. Hence it is imperative to purge some of those that have become irrelevant and renew ourselves to stay current. Given that technologies, process, domain knowledge are constantly changing every day, it is necessary to refine what we know by purging those that have become irrelevant and acquire newer ones.

The key change element is the mindset. We need to give up on a static mindset. Refining perceptions, the way we think, respond, act need to change. Looking at history and stating 'this is how it was done all these years' could make us a dinosaur.

A fresh mind that does not dwell only in the past, but adapts/adjusts to the current for a better future is very necessary.

Lastly to give up requires us to learn new things. Most often we are very happy to look at our world of testing/technology/business and expand our knowledge. It is necessary to 'give up' seeking knowledge only from our disciplines. Go outside the domains and read/watch/listen about anything. You never know where inspiration will come from, where new ideas will emerge from. Inspirations and new ideas are the bedrock for growth, to do new things, not just more. To deliver higher value, not just do faster and better.

As you step into another wonderful year, reflect on things you have done, ask what you are going to give up, to be able to accomplish new things, to grow. Make time to do more/better work, renew knowledge to do new work, adapt your thinking to be able to do newer work and finally be inspired from other disciplines. It is not just about 'adding more', it is really about 'letting go'.

Wish you a wonderful 2016!

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

testomaton

Quality Assurance via Automation

Software testing startup specializing in test automation services, consulting & training for QA teams on how to build and evolve automation testing suites.

Services

Test System Analysis and work estimation

Review of client's system (either in development or on early stage) to produce a Test Plan document with needed test cases and scenarios for automation testing.

Tests creation and Automation

Development of test cases (in a test plan) and Test Scripts to execute the test cases.

Regression and Test evolution

Development of Regression test suites and New Features suites, including test plans and test scripts or maintenance of existing.

Architecture Consulting

Review of software architecture, components and integration with other systems (if applicable).

Trainings

Depending on the particular need, we can provide training services for: Quality Assurance theory and best practices, Java, Spring, Continuous Integration, Groovy, Selenium and frameworks for QA. For further information please check our web site.

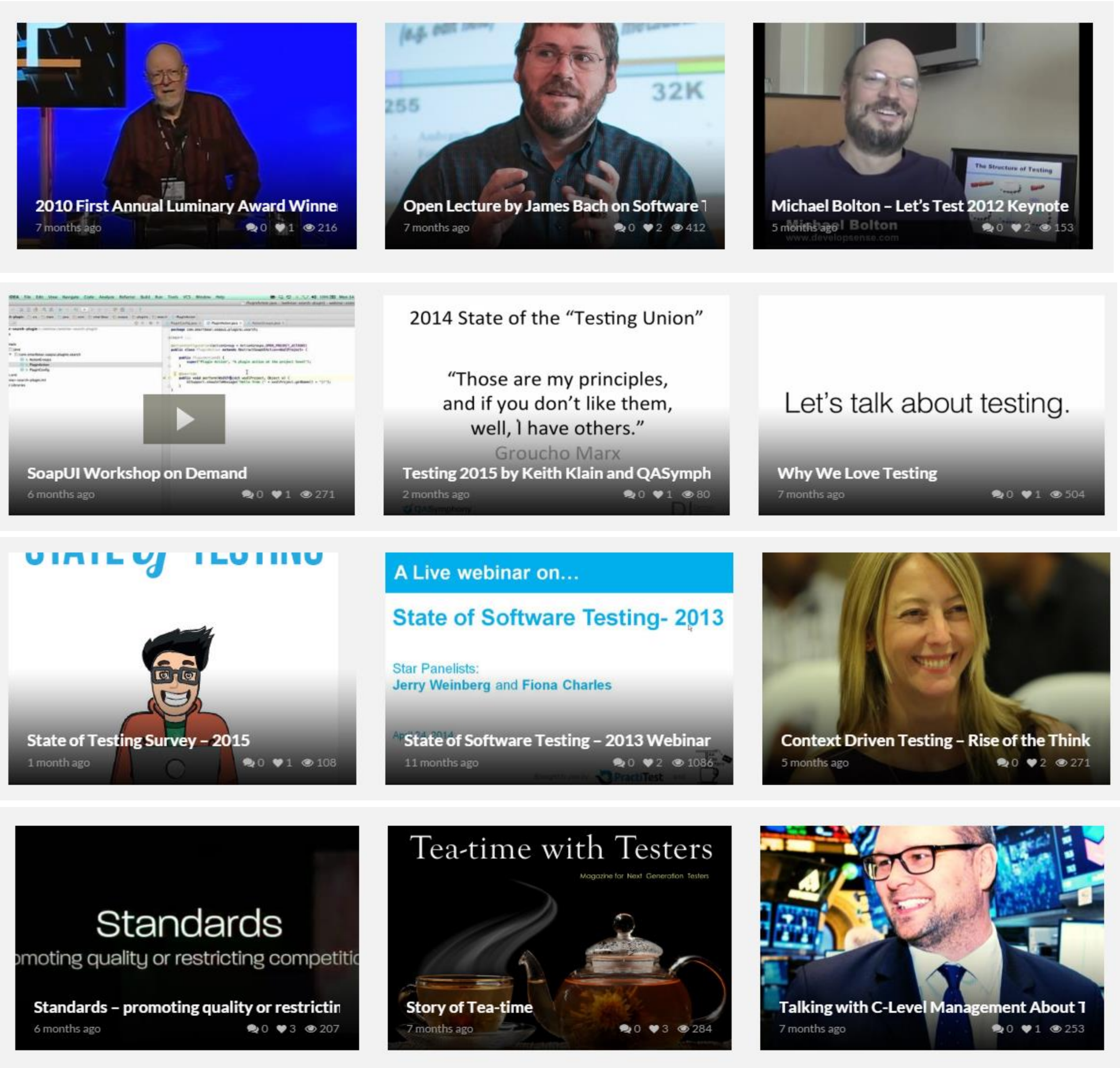
We enjoy putting our experience to your service. Our know-how allows us to provide consultation services for projects at any stage, from small up to super-large. Due to our range of expertise we can assist in software architecture and COTS selection; review of software designs; creation of testing suites and test plans with focus on automation; help building quality assurance teams via our extensive training curriculum.

look us up: www.testomaton.com - twitter: @testomaton

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

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

Why We Love Testing
7 months ago 💬 0 ♥ 1 👁 504

State of Testing Survey – 2015
1 month ago 💬 0 ♥ 1 👁 108

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 restrictin
6 months ago 💬 0 ♥ 3 👁 207

Story of Tea-time
7 months ago 💬 0 ♥ 3 👁 284

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

32K
The Structure of Testing
Michael Bolton
www.developsense.com
2014 State of the "Testing Union"
"Those are my principles, and if you don't like them, well, I have others."
Groucho Marx
Let's talk about testing.
STATE of TESTING
A Live webinar on…
State of Software Testing- 2013
Star Panelists:
Jerry Weinberg and Fiona Charles
PractiTest
Standards
promoting quality or restricting competitio
Tea-time with Testers
Magazine for Next Generation Testers

WWW.TVFORTESTERS.COM

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…

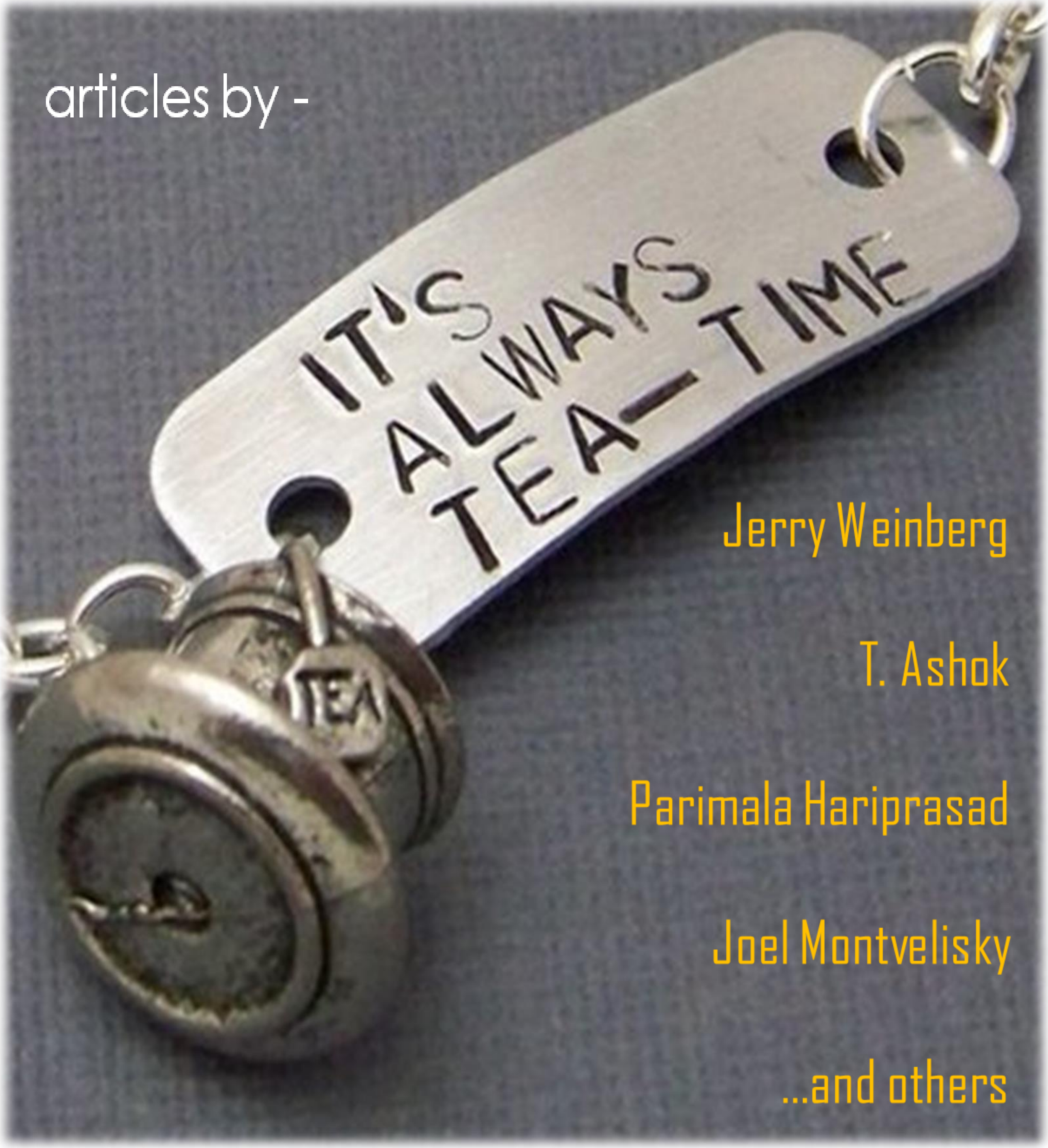
Now we have some more awesome offerings at pretty affordable prices.

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

Parimala Hariprasad

Joel Montvelisky

...and others

our family

Founder & Editor:

Lalitkumar Bhamare (Pune, India)

Pratikkumar Patel (Mumbai, India)

Lalitkumar

Pratikkumar

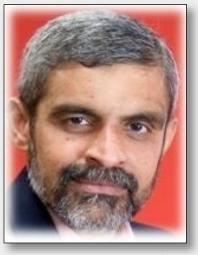
Contribution and Guidance:

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 – Ali Express – Life hall today

Core Team:

Dr.Meeta Prakash (Bangalore, India)

Unmesh Gundecha (Pune,India)

Dr. Meeta Prakash

Unmesh Gundecha

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