THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

YEAR 6 ISSUE VI

Articles by –

Jerry Weinberg

John Stevenson

Sunjeet Khokhar

T Ashok

Maaike Brinkhof

Viktor Slavchev

Giuseppe Cilia

Over a Cup of Tea with Mike Kelly

AGILE TESTING DAYS

EUROPE´S GREATEST AGILE SOFTWARE TESTING EVENT!

DECEMBER 05–09, 2016, POTSDAM, GERMANY

BEST AGILE RELATED CONFERNCE IN EUROPE
2ND BEST IN THE WORLD

MORE THAN 40 NATIONS

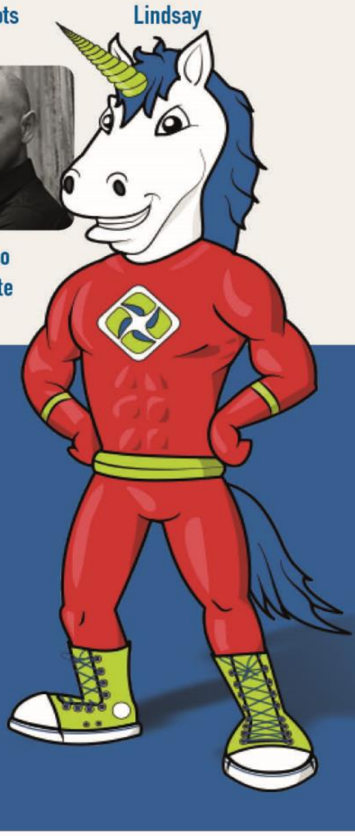
1 UNCONFERENCE DAY, 2 TUTORIAL DAYS, 3 CONFERENCE DAYS

11 KEYNOTES, 12 TUTORIALS, 100 SESSIONS OF TALKS AND WORKSHOPS, LOTS OF BONUS SESSIONS & DAILY SOCIAL EVENTS

THIS YEARS KEYNOTE SPEAKERS

Abby Fichtner, Alexandra Schladebeck, Bart Knaack, Diana Larsen, Gojko Adzic, Huib Schoots, James Lindsay

Jessica DeVita, Keith Klain, Melissa Perri, Michael Wansley, Mike Sutton, Vasco Duarte



Save yourself some money and get your ticket with 10% off.
Just enter following code in to your registration under:
www.agiletestingdays.com/registration

Discount-Code:

TeaTimeTester_010

For more infos, visit **www.agiletestingdays.com**

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

Rising in the Fall

While enjoying my morning tea, I stumbled upon this lovely poem by Maya Angelou:

Carefully , the leaves of autumn

sprinkle down the tinny sound of little dyings

and skies sated of ruddy sunsets of roseate dawns

roil ceaselessly in cobweb greys and turn

to black for comfort.

Only lovers see the fall

a signal end to endings a gruffish gesture alerting

those who will not be alarmed that we begin to stop

in order to begin again.

My apologies for the terrible delay we have had in bringing you new issues. The times at personal and professional front had been demanding and everything else took the back-seat. But guess what, just like each season comes with its own flavor, such times in a year have pleasure of their own. I did enjoy those busy days and I am now enjoying the colorful autumn too.

As Maya aptly says in her poem, it's about stopping in order to begin again. Here is to our beginning again…here is to the rising in the fall.

See you soon with another colorful issue. Until then…

Sincerely Yours,

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

[illegible]

QuickLook

OnlineTestConf
The Online Testing Conference
THE FIRST 100% ONLINE TESTING CONFERENCE!
November 29-30, 2016

AGILE TESTING DAYS
EUROPE'S GREATEST AGILE SOFTWARE TESTING EVENT!
December 05–09, 2016
Potsdam, Germany
DISCOUNT-CODE TeaTimeTester_010

LST News

Backdoored Pokemon GO App Infects Android Devices

Source: Security Week

Popular mobile games represent a productive attack vector for cybercriminals, and the Pokémon GO augmented reality Android game released last week is the most recent proof of that.

The first Pokémon game sanctioned by Nintendo for iOS and Android devices was released in Australia and New Zealand on July 4 and landed in the US on July 6, but the rest of the world hasn't received it via official channels. Three days after arriving in the US, Pokémon GO became one of the most used apps in the Google Play Store, SimilarWeb data reveals.

For cybercriminals, this represented a great opportunity and they were fast to take advantage of it: a modified Pokémon GO APK packing the malicious remote access tool (RAT) called DroidJack was spotted less than 72 hours after the game was officially released. The targets of this malicious game were users outside those three geographies, which were expected to head to third-party portals to grab it.

It's not uncommon for users to turn to third parties to grab an application or game unavailable in their area, especially when many publications provide details on how the side-loading can be done. However, apps downloaded from unofficial portals often carry hidden risks, the main reason for which users are always warned against this practice.

In the case of Pokémon GO, the attackers were very quick about it: they created a malicious APK within three days after the initial launch, taking advantage of the hype surrounding the official game. However, those installing this program from a third-party might have been warned of its malicious intent if they paid close attention to the requested permissions.

The DroidJack (also known as SandroRAT) malware hidden within the APK requested some unusual permissions during installation, researchers at Proofpoint explain. These include permissions to read and edit text messages, make phone calls, record audio, modify contacts, read bookmarks and web history, connect to Wi-Fi, and to retrieve running apps at startup.

All of these permissions fall in line with the functionality previously associated with DroidJack, a mobile threat that has been around since 2014. The Trojan can steal user messages, call logs, contacts, browser history, and installed apps, and can also execute remote commands such as take photos, record videos and calls, send SMS, and more.

Released in Google Play as Sandroid in 2013, DroidJack was initially designed as a legitimate app that allowed users to control their PC from an Android device. SandroRAT first emerged in December 2013 on a hacker forum, but the DroidJack variant was announced only in June 2014. It was offered on its own site at $210 for a lifetime package, Symantec researchers revealed in Novermber 2014.

In October 2015, European law enforcement agencies staged a coordinated swoop on suspected users of DroidJack which bought the malware and used it in 2014 and 2015. In November 2015, researchers analyzed OmniRAT, an Android tool similar to DroidJack in that it was initially designed as a legitimate application for remotely controlling Android devices, but later became malicious.

According to Proofpoint, the Pokémon GO game was modified in a manner meant to deceive users into believing they have installed the real game, and both versions feature the same start screen. The good news is that the APK wasn't observed in the wild, although the researchers did notice it in a malicious file repository service.

The security researchers also explain that the DroidJack RAT has been configured to communicate to the command and control (C&C) domain pokemon[.]no-ip[.]org over TCP and UDP port 1337. The C&C domain resolved to an IP address in Turkey (88.233.178[.]130), researchers say, adding that the IP was not accepting connections from infected devices at the time of the analysis.

Although the infected Pokémon GO APK wasn't observed in live attacks, it represents the perfect example of why users should always download applications only from trusted sources. Cybercriminals keep a close eye on trending applications and games and will definitely prey on their popularity to carry out their nefarious activities.

"Installing apps from third-party sources, other than officially vetted and sanctioned corporate app stores, is never advisable. Official and enterprise app stores have procedures and algorithms for vetting the security of mobile applications, while side-loading apps from other, often questionable sources, exposes users and their mobile devices to a variety of malware," Proofpoint researchers say.

INTERVIEW **PART II**

In the community of thinking testers, Mike Kelly is a familiar name. He is the creator of FCC CUTS VIDS heuristic, popularly known as Touring Heuristic.

Currently, Mike is looking after DeveloperTown as its Managing Partner among other interesting things he does.

Interviewing Mike has been on our wish-list and I'm glad that we could finally do it. Special thanks to Dirk for pairing up with me on this task.

Mike has provided insightful answers to our questions and for the benefit of our readers, we are publishing his interview in two parts rather than editing/shortening the answers to fit in one single issue.

I'm sure that you'll like what we discussed.

- Lalitkumar Bhamare

Over A Cup of Tea with Mike Kelly

Apart from software testing, what are the areas within software development that fascinate you? Why?

I'm currently fascinated by early customer development for a product. It's like this awesome mash-up of sales, marketing, analytics, and testing.

At Tenant Tracker, we built the first version of the product for one specific customer. That's a relatively straightforward process. You work with that one customer to figure out what they need, and you basically build it for them – with a slightly longer view then they might have for what the rest of the market might want.

Once that first client was live, the Tenant Tracker team started getting to work on selling clients two and three. Client's two and three are interesting when it comes to feature requests. You can't tell them what you told client one – "This solution will start off custom tailored to your business." For clients two and three, it's more like, "Tell us what features are critical to you. There's a good chance we have them already, but if not we will work with you to define and build them." But now you need to mash the needs of those new clients with the needs of the first client.

If the needs of those customers conflict, who wins? That's where marketing enters the picture. You have to try to guess what the nameless/faceless "market" wants. You know, all those potential future customers. Your marketing team (or you if you're a startup) now has to go research the heck out of existing market to try to figure out which early adoption customer wins. You have to make the call based on what you think the broader market will want. And maybe you need to design and run some early market validation exercises like: A/B testing marketing site conversions, testing pay-per-click campaigns, market surveys, prospect interviews, etc.

At Tenant Tracker, once we had clients two and three on board, we now had a new problem. We were no longer selling clients one at a time. Now we were trying to sell and onboard multiple clients at one time. It's at this point the early team loses the ability to start selling "customization" as a part of the deal. Your crazy small development team of two people can't juggle new feature development, client on-boarding, bug fix, production support, and sales commitments for the next eight prospects. So what do you do now?

Now you start to do "real" product ownership. You start to build a customer request backlog. When a client or prospect requests a feature, you say "Thank you so much. You're not the first to ask for that. We will add it to the backlog and let you know when we think we can address it." At each subsequent sprint planning, the sales guy argues for the features the current largest prospect wants, the lead developer argues for paying down the current largest piece of technical debt that's going to affect scale, the marketing guy argues for the best feature that will attract more prospects, the analytics guru shares what features customers are actually using, and the support guy argues for bug fixes and "small" client requests that should "only take a couple hours."

That's where we are on Tenant Tracker. Trying to juggle early customers, marketing data, sales data, and our vision for where the product should go. It's awesome! And this is what I think most testers on small teams get to do daily. It's all about triage, data-driven decision making, and communication. The hardest part is making sure you have a team that's committed to following through on the decisions that get made. (The Tenant Tracker team is above average at that I think.)

There are a ton of small strategies to juggling those tensions well. I feel like the exposure I've had to seeing many of our DeveloperTown clients wrestle with this have been invaluable. I have a front row seat to see what works and what doesn't. I can use that data to refine my models of what might work for early sales strategies for certain types of products. I have a list of the critical risks related to balancing customer expectations, technical debt, and revenue. And I keep an ever growing list of market validation techniques.

We learned that you also have a passion for farming. Do you see any similarities between your passion for farming and software quality?

About four years ago I started a hobby farm. We currently raise chickens, turkey, and bees. We have a fairly large garden. And we briefly experimented with quail. (My wife voted them off the island.)

Software testing is a never-ending process. There are an infinite number of tests you could run. As testers, we're paid to think through risk and coverage, so we can (hopefully) determine the most optimal set of tests to run in the time we're given. Farming is a lot like that. There's always more you can do on a farm: something to feed, something to fix, wood to chop, weeds to pull, etc.

You quickly recognize that if you want to have a life outside of farming, you need to assess risk and coverage. What's the important stuff, and how much of it do I need to do if I want a "good enough" farm? Do I care more about how my farm looks? Or do I care more about the health of the animals? Am I trying to make money? Or am I using this as an outlet to "recharge"? It's all tradeoffs.

In the last four years I've picked up hundreds of tools. I'm now an okay carpenter. I can administer first aid to poultry. I can make my own sausage (literally from scratch). I'm figuring out how to preserve food. I've picked up hundreds of little skills, but I'm not the master of any of them. I know just enough to get by. That's exactly like software testing. Testers know a little about a lot. We learn enough to get the job done, and then move on to the next item in the list.

I think farmers have it a bit easier in that the goals/priorities are (typically) clearer. The animals need to stay alive. Ideally we'd like them to be comfortable and happy while they are alive. And ideally, we'd like to make a profit at the end of the summer. If only the goals of a software tester were so simple. :)

In present day scenario, what are the testing skills you think testers must possess?

Before we look for testing skill, we look for ownership, someone who doesn't take themselves too seriously, an interest in learning new things, comfort with making critical decisions with little information, the ability to ask insightful questions, and good communication. That quickly gets us down to a smaller handful of candidates. From there, we start to look at specific skills and experiences.

To rapid-fire some quick critical skills that I think we look for:

- the ability to model a product quickly;
- the ability to extract requirements verbally;
- clarity in writing up and articulating issues;
- the ability to prioritize work;
- and the ability to assess risk.

Let's just look at those last two: prioritization and risk assessment. If you wanted to identify those skills in someone, you might ask them to look at ten defects in a backlog and see if they can quickly provide input regarding which ones likely matter and which the team might be able to safely ignore for some period of time. Their answer might not align perfectly with the product owner, but you get a feel for how they assess risk by what types of questions they ask during the assessment. And you can then see how they prioritize work based on that understanding of risk.

I think we favor testers who have some level of technical background (code, networking, etc.). We find that it makes them more insightful in terms of questioning, and allows them to come up to speed faster when faced with new technology. And I think we also favor testers who have some experience going deep on a product. While we rarely get the chance to go deep ourselves, the experience you can gain by spending a lot of time on one project is unique, and colors how you think about testing a product over time.

Are there any interesting stories from your coaching experience that you would like to share with us?

All of my coaching these days is around the core business. What's the problem you're trying to solve? Do you have the right team? Is the market big enough? Does the business model work on paper? How do we set prices? Do you plan on fundraising or bootstrapping? Do you know what it's actually going to cost to develop the product for the next three to five years? Where does testing fit into all of that? Sometimes it's tactical - "You need to add a tester to your financial model." Sometimes it's strategic – "What if we added a feature that allowed our users to perform quality control on the data without them even knowing it?" And many times it's educational - "When you say things like, 'It works on all Android devices…' you know what that means, right?"

Here's a current example. I'm working with a founder right now who wants to launch a startup with a two-sided marketplace. An example of a two-sided marketplace is eBay. For eBay to work, you need users willing to buy and users willing to sell. This is the scariest of all startups. You have two separate marketing and customer acquisition campaigns going on at one time. It's so hard to do because sellers don't want to enter a market when there aren't buyers already there. And buyers go to the site one time, don't see any sellers and then they never go back. In many startup circles, this is called the "chicken and the egg problem."

It's solvable, but normally it's solved with capital. You just raise enough money that you can afford an overwhelming campaign focused on one side of the market in an effort to get a chicken. Then you can worry about eggs. Sometimes it's solved with elegance. Where the startup launches as a single-sided market, and pivots into a two-sided marketplace down the road. It makes that pivot once it already has a bunch of eggs.

(Okay… I'll stop with that metaphor. Sorry.)

With the startup I'm actively coaching, we're exploring options. Things like:

- What are the value propositions that most resonate with your buyers?
- What are the value propositions that most resonate with your sellers?
- If you were to offer your buyers something else – not your original product – that provided a similar value, what could that look like?
- If you were to offer your sellers something else – not your original product – that provided a similar value, what could that look like?
- Would your buyers pay for that other thing? Even if there were no sellers yet? How much would they pay?
- Would your sellers pay for that other thing? Even if there were no buyers yet? How much would they pay?

Each of those questions might get us zero to ten ideas. Our job is to now test those ideas. We need to validate that our assumptions are correct, and we need to see if any of these ideas actually have traction in the market. If they do – then that idea might be the key to unlocking the two-sided marketplace problem.

Coaching becomes an exercise in educating the founder how to do those tests. What are all the ways we can reach out potential users to validate our assumptions (knowledge of testing techniques)? How do you identify your target market (coverage)? What do we ask them (test design)? How will we know when we're done (coverage, statistics, stopping heuristics)? When we get an answer, how should we interpret it (modeling)? What tools should we use? When should we run the test? How do we want to track results? Maybe we do some of that work, but most of the time we like to have the founder engaged in these activities. It's one thing to have someone else run a test for you and give you the results summary, and entirely another to run that test yourself and see/hear/feel the feedback directly. Are your potential customers passionate about the problem the way you are? Do they really want it to exist? Or is it just a nice to have? Is there something special about this prospect that could inform how I target future prospects? All of those are hard to pick out of a report. You need that direct experience.

The language we use is "market validation." But it's testing. A lot of the lean startup methodology is test-first, build second. My experience as a tester bleeds into a lot of my current coaching work – even though I'm not coaching testers.

From the recommendations you make on your blog, it's visible that you have read lots of books on multiple disciplines. What books/resources would you recommend for test professionals for their further learning?

There are a TON of places testers can go to find recommendations on testing books. I don't want to repeat that here. Instead, I'll offer some gems that I've found that I think can help testers raise their game in some specific areas:

- Want people to take you seriously when you say something? Read "Extreme Ownership" by Willink/Babin and "Pitch Anything" by Klaff. The Extreme Ownership chapters Believe and Prioritize and Execute will be particularly useful to testers.
- Want to build a team that truly delivers ridiculous results? Read "Creativity Inc." by Catmull/Wallace, "Drive" by Pink, and "Cleaning House" by Wyma. Which one of these is not like the others? Cleaning House is about raising your kids. It's dead on for crushing entitlement in your team as well. Read it through the filter of your work life.
- Want to change the conversation about how your organization should approach software testing? Read "Antifragile" by Taleb and "Good Strategy / Bad Strategy" by Rumelt. All of Teleb's books should be required reading for software testers, but Antifragile is the "capstone." Read it.
- Want to change the way you think about offshoring? Read "The Lexus and the Olive Tree" by Friedman and "The 4-Hour Workweek" by Ferriss. Each will challenge you in a subtle way to change the way you think about the future of how software will be developed – and what your role in that might look like.
- Struggling with depression, motivation, or a stressful work situation (like a startup)? Read "Resilience" by Greitens. This was my favorite book of 2015. It's exactly what I needed while struggling with some serious depression.
- Not into all that non-fiction and want a fun read that might teach you how to be a better tester? Pickup any Sherlock Holmes anthology and start readying. (The books are better than all the TV shows – and that's saying something.) I also recommend the "Gentleman Bastard" series by Lynch. Each book is themed a bit differently, but I certainly get an Arthur Conan Doyle vibe when I read them.

What is the next cool thing we can expect from Mike Kelly in coming future? (You may want to talk about some ongoing project or so…)

Well, probably nothing specific to software testing. At DeveloperTown, we're excited to continue growing as a consulting business, and we're launching a couple of new companies/initiatives in the next few months. If we are successful, over the next five years you'll see our brand grow nationally, and we will have more than just one of our own startups out in the market. While it's crazy what we've done in six years, we still see a ton of potential in front of us for what else we can do.

Personally, I'm excited to keep learning more about investing in early-stage startups, each year I get a deeper appreciation for what it takes to scale a business, and I'm constantly learning (mostly the hard way) better ways to lead. Maybe someday I'll get back to blogging about some of those experiences, but it's really difficult to find the time. Writing has always helped me firm up my ideas. Even this interview has helped me organize my thoughts around certain topics. So maybe this is a catalyst to get me writing again. That would be cool.

What would be your general advice for our readers who are mostly test professionals?

If you're not already, become a leader on your team. Leadership isn't about positional authority; it's about knowledge and experience, ownership for your work, caring for the people around you, and caring about outcomes. To be a leader, you don't have to be the best tester, the best coder, or the best anything else. You just need to be competent, drive things forward, and be the person the rest of the team can rely on.

When you think back to the teams you've worked with over the years, who stands out in your mind? I'm willing to bet that if it's more than five years ago, it's someone who was a leader on that team or in that organization. It wasn't the person with the most technical knowledge. It was likely the person who drove efforts forward. They likely didn't complain all that often. They didn't make excuses if they missed a deadline. They likely took a little risk now and then – spending some personal capital to try to create a better outcome for the team around them. They likely coached better performance out of you and others.

If that's not you on your current team, why not? In general, as an industry we need more leaders. In software testing, we definitely need more leaders. We're in a perfect support role for the rest of the team around us. Most testers touch every aspect of the product and get involved in almost all phases of a project. If you're great a testing (modeling, asking questions, critical thinking, etc.) then you likely have all the skills you need to be a great leader. Make that jump.

All of my opportunities came from stepping up and leading. Early on I wasn't asked to lead – I just did it. Up until DeveloperTown, my ability to influence was always based on my street cred – it was never based on my positional authority. Even here at DeveloperTown, I think a lot of my influence has to do with credibility that comes from doing – not talking. (I'm sure I'm wrong about that… my role probably has more direct influence than I'd like.)

Be someone who gets stuff done. Help others get stuff done. Make your team successful. If you do that, you'll always be in demand as a software tester.

Our last question for today, do you read Tea-time with Testers? We'd love to know your feedback.

I'm sad to say that I don't read it on a regular basis. I still follow a number of testers on social media, so if I see something talked about I'll read it. But most of my reading these days goes toward topics where I have much less experience.

That said, if any of your readers didn't read your Kaner's interview in your September 2015 issue, they should go back and find it. Kaner's insights into how the "state of testing" hasn't progressed are dead on.

Preventing a Software Quality Crisis

Abstract

Many software development organizations today are so overloaded with quality problems that they are no longer coping with their business of developing software. They display all the classic symptoms of overloaded organizations—lack of problem awareness, lack of self-awareness, plus characteristic behavior patterns and feelings. Management may not recognize the relationship between this overload and quality problems stemming from larger, more complex systems. If not, their actions tend to be counterproductive. In order to cure or prevent such a crisis, management needs to understand the system dynamics of quality.

Symptoms of Overload Due to Poor Quality

In our consulting work, we are often called upon to rescue software development operations that have somehow gotten out of control. The organization seems to have slipped into a constant state of crisis, but management cannot seem to pin the symptoms down to one central cause. Quite often, that central cause turns out to be overload due to lack of software quality, and lack of software quality due to overload.

Our first job as consultants is to study symptoms. We classify symptoms of overload into four general categories—lack of problem awareness, lack of self-awareness, plus characteristic patterns of behavior and feelings. Before we describe the dynamics underlying these symptoms, let's look at some of them as they may be manifest in a typical, composite organization, which we shall call the XYZ Corporation.

Lack of Problem Awareness

All organizations have problems, but the overloaded organization doesn't have time to define those problems, and thus has little chance of solving them:

1. Nobody knows what's really happening to them.

2. Many people are not even aware that there is a system-wide problem.

3. Some people realize that there is a problem, but think it is confined to their operation.

4. Some people realize that there is a problem, but think it is confined to somebody else's operation.

5. Quality means meeting specifications. An organization that is experiencing serious quality problems may ignore those problems by a strategy of changing specifications to fit what they actually happen to produce. They can then believe that they are "meeting specifications." They may minimize parts of the specification, saying that it's not really important that they be done just that way. Carried to an extreme, this attitude leads to ignoring certain parts of the specification altogether. Where they can't be ignored, they are often simply forgotten.

6. Another way of dealing with the overload is to ignore quality problems that arise, rather than handling them on the spot, or at least recording them so others will handle them. This attitude is symptomatic of an organization that needs a top-to-bottom retraining in quality.

Lack of Self-Awareness

Even when an organization is submerged in problems, it can recover if the people in the organization are able to step back and get a look at themselves, in the chronically overloaded organization, people no longer have the means to do this. They are ignorant of their condition, and they have crippled their means of removing their ignorance:

7. Worse than not knowing what is going on is thinking you know, when you don't, and acting on it. Many managers at XYZ believe they have a grip on what's going on, but are too overloaded to actually check. When the reality is investigated, these managers often turn out to be wrongs For instance, when quizzed about testing methods used by their employees, most managers seriously overestimate the quality of testing, when compared with the programmers' and testers' reports.

8. In XYZ) as in all overloaded organizations, communication within and across levels is unreliable and slow. Requests for one kind of information produce something else, or nothing at all. In attempting to speed up the work, people fail to take time to listen to one another, to write things down, or to follow through on requested actions.

9. Many individuals at XYZ are trying to reduce their overload by isolating themselves from their co-workers, either physically or emotionally. Some managers have encouraged their workers to take this approach, instructing them to solve problems by themselves, so as not to bother other people.

10. Perhaps the most dangerous overload reaction we observed was the tendency of people at XYZ to cut themselves off from any source of information that might make them aware of how bad the overload really is. The instant reaction to any new piece of information is to deny it, saying there are no facts to substantiate it. But no facts can be produced because the management has studiously avoided building or maintaining information systems that could contradict their claims that, "We just know what's going on." They don't

know, they don't know they don't know, and they don't want to know they don't know. They're simply too busy.

Typical Behavior Patterns

In order to recognize overload, managers don't have to read people's minds. They can simply observe certain characteristic things they do:

11. The first clear fact that demonstrates overload is the poor quality of the products being developed. Although it's possible to deny this poor quality when no measurements are made of the quality of work in progress, products already delivered have shown this poor quality in an undeniable way.

12. All over the organization, people are trying to save time by short-circuiting those procedures that do exist. 'This tactic may occasionally work in a normal organization faced with a short-term crisis, but in XYZ, it has been going on for so long it has become part of standard operating procedure.

13. Most people are juggling many things at one time, and thus adding coordination time to their overload. In the absence of clear directives on what must be done first, people are free to make their own choices. Since they are generally unaware of the overall goals of the organization, they tend to sub-optimize, choosing whatever looks good to them at the moment.

14. In order to get some feeling of accomplishment, when people have a choice of tasks to do, they tend to choose the easiest task first, so as to "do something." This decision process gives a short-term feeling of relief, but in the long term results in an accumulation of harder and harder problems.

15. Another way an individual can relieve overload is by passing problems to other people. As a result, problems don't get solved, they merely circulate. Some have been circulating for many months.

16. Perhaps the easiest way to recognize an overloaded organization is by noticing how frequently you hear People say, in effect, that they recognize that the way they're working is wrong, but they "have no time to do it right." This seems almost to be the motto of the XYZ organization.

Typical Feelings

If you wait for measurable results of overload, it may be too late. But it's possible to recognize an overloaded organization through various expressions of peoples' feelings:

17. An easy way to recognize an overloaded organization is by the general atmosphere in the workplace. In many areas at XYZ there was no enthusiasm, no commitment, and no intensity. People were going through the motions of working, with no hope of really accomplishing their tasks.

18. Another internal symptom of overload is the number of times people expressed the wish that somehow the problem would just go away. Maybe the big customer will cancel the contract. Maybe the management will just slip the schedules by a year. Maybe the sales force will stop taking more orders. Maybe the company will fail and be purchased by a larger company.

19. One common way of wishing the problem would go away is to choose a scapegoat, who is the personified source of all the difficulties, and then wish that this person would get transferred, get fired, get sick, or quit. At XYZ, there are at least ten different scapegoats—some of whom have long gone, although the problems still remain.

20. Perhaps the ultimate emotional reaction to overload is the intense desire to run away. When there are easy alternatives for employees, overload is followed by people leaving the organization, which only increases the overload. The most perceptive ones usually leave first. When there are few attractive opportunities outside, as at XYZ, then people "run away" on the job. They fantasize about other jobs, other places, other activities, though they don't act on their fantasies. Their bodies remain, but their hearts do not.

The Software Dynamics of Overload

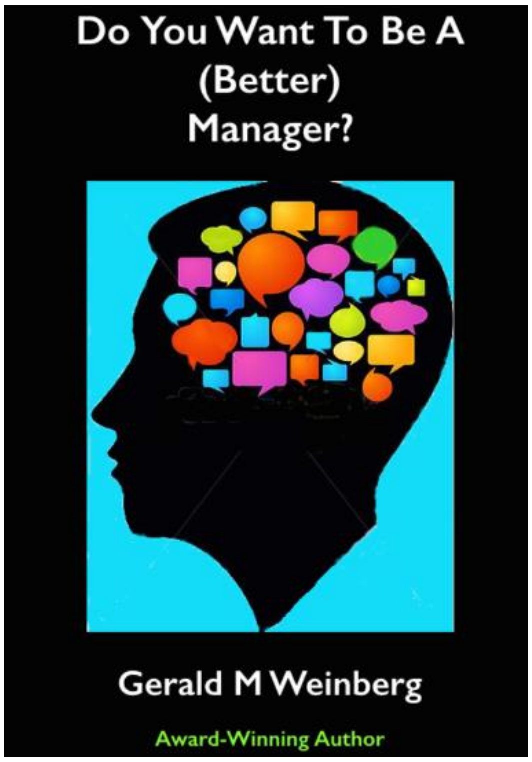
There are a number of reasons for the overload situation at organizations like XYZ, but underlying everything is the quality problem, which in turn arises from the changing size and complexity of the work. This means that simple-minded solutions like adding large numbers of people will merely make the problems worse. In order for management to create a manageable organization, they will have to understand the dynamics of quality. In particular, they will have to understand how quality deteriorates, and how it has deteriorated in their organization over the years. The XYZ Company makes an excellent case study.

The quality deterioration at XYZ has been a gradual effect that has crept up unnoticed as the size and complexity of systems has increased. The major management mistake has been lack of awareness of software dynamics, and the need for measurement if such creeping deterioration is to be prevented.

The quality deterioration experienced at XYZ is quite a common phenomenon in the software industry today, because management seems to make the same mistakes everywhere—they assume that the processes that would produce quality small systems will also produce quality large systems. But the difficulty of producing quality systems is exponentially related to system size and complexity, so old solutions quickly become inadequate. These dynamics have been studied by a number of software researchers, but it is not necessary to go fully into them here. A few examples will suffice to illustrate specifically what has been happening at XYZ and the kind of actions that are needed to reverse the situation.

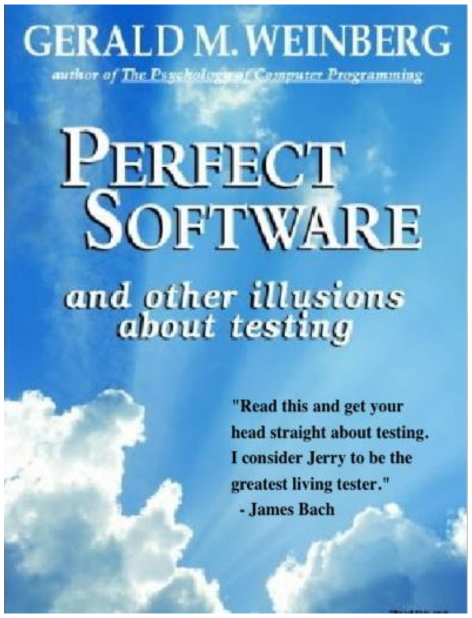
NOTE: The remaining two-thirds of this article describes these dynamics and corrective actions, and can be found as a new chapter in the book, **Do You Want To Be A (Better) Manager?**

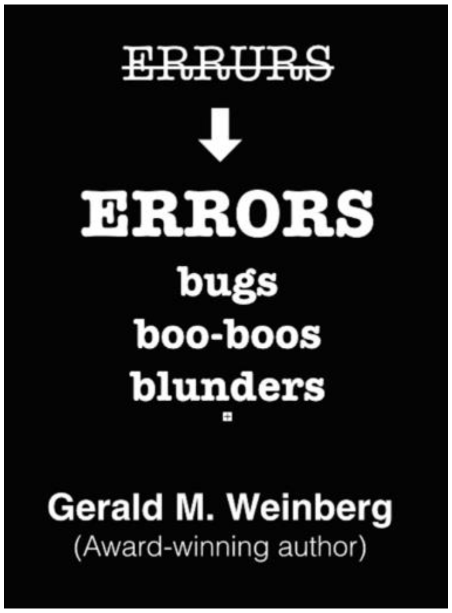
The book also details a number of the most common and distressing management problems, along with dozens of positive responses available to competent managers.



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.

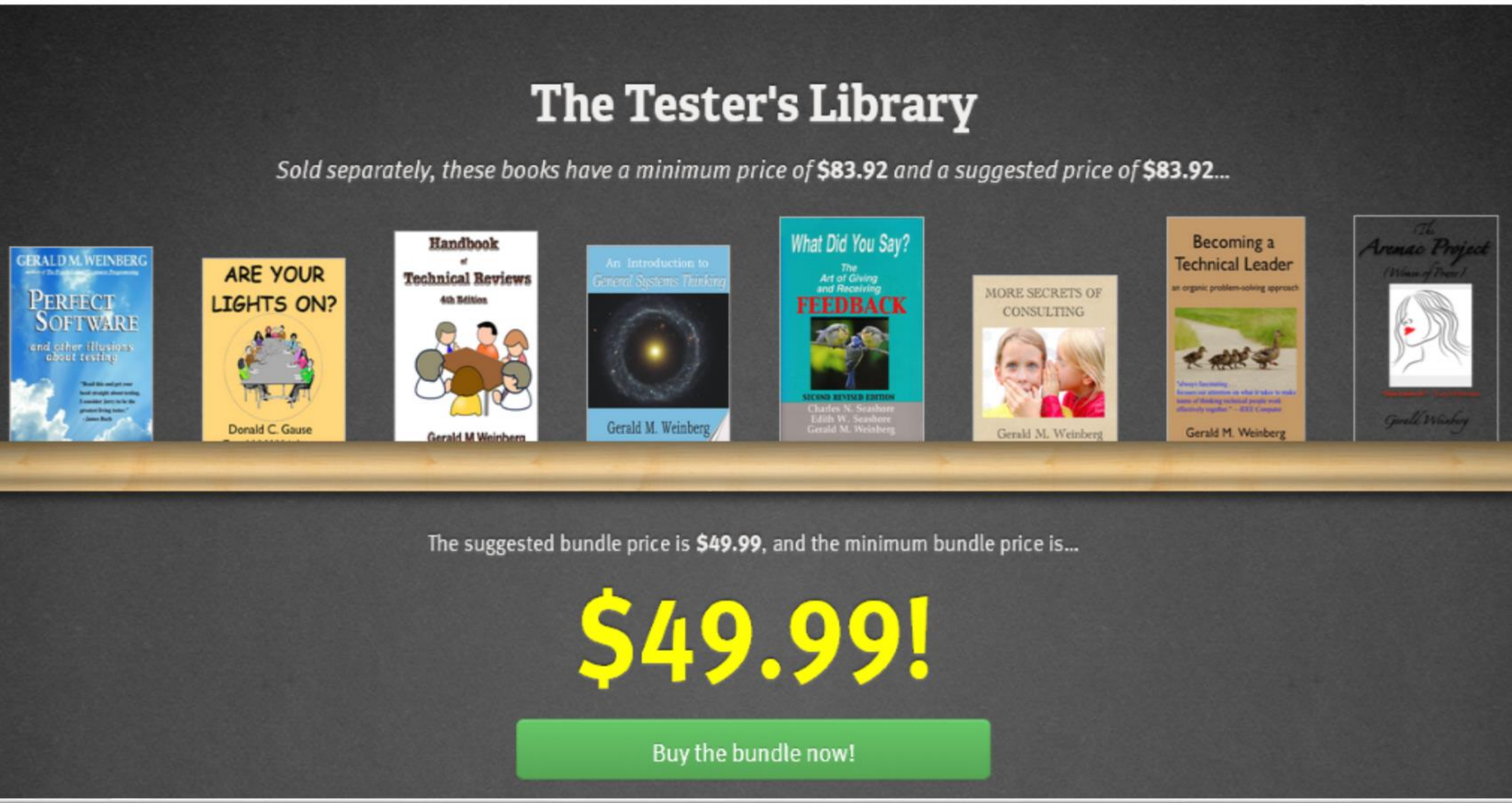
Lately, he's been observing the Agile movement, and he's offering an evolving set of impressions of where it came from, where it is now, and where it's going. If you are doing things Agile way or are curious about it, this book is for you.

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...

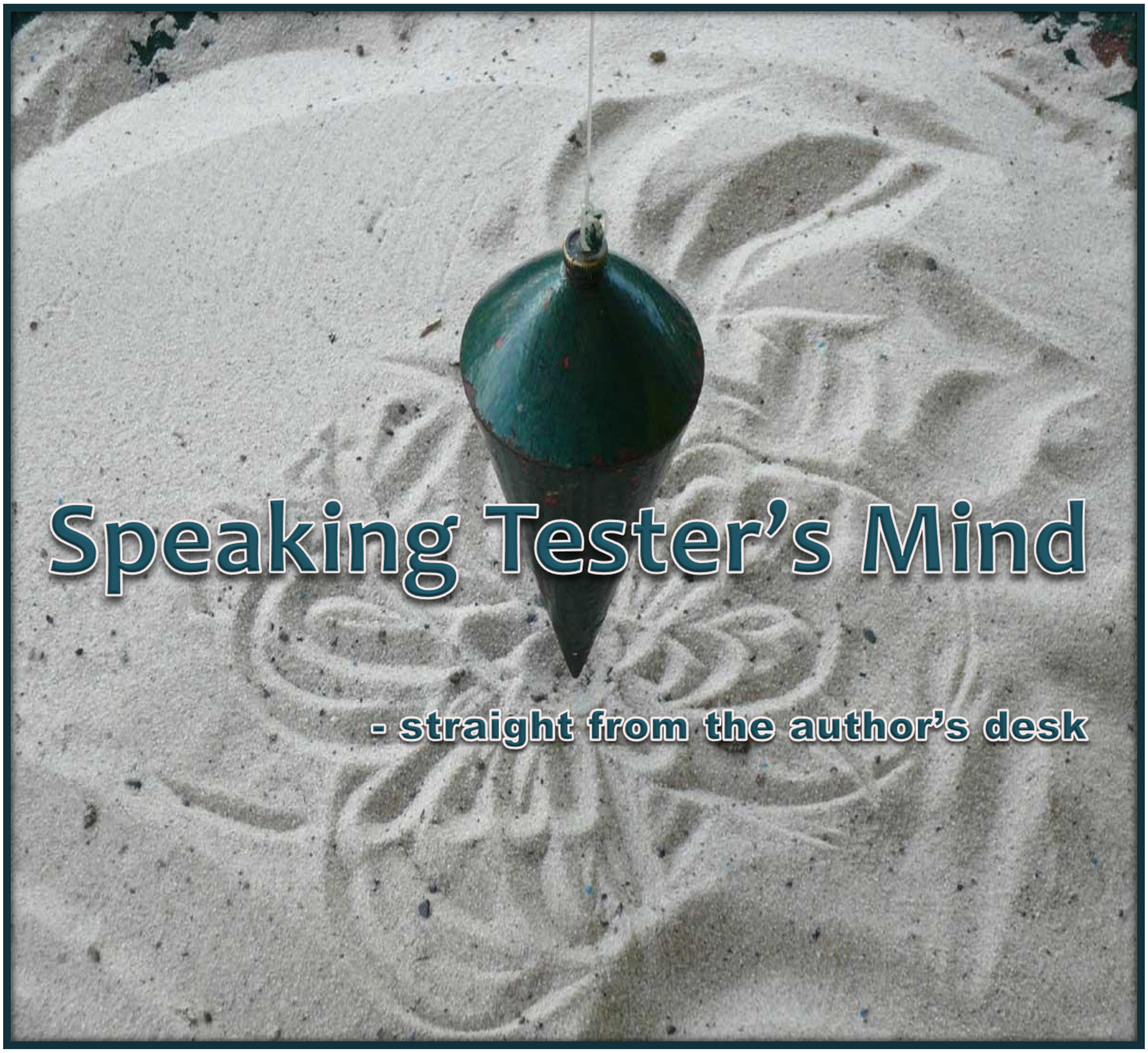
$49.99!

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 6
Speaking the Language of Business

- by John Stevenson

This article is based on a talk given by Keith Klain at both CAST and Testbash

- Talking to C Level Management about testing - https://www.youtube.com/watch?v=Xn-2iIs-VBA – CAST
- How to talk to your CIO if you really have to https://www.youtube.com/watch?v=CurOi7jKJ1M - Testbash

As testers we find it difficult to explain our value when we get given opportunities to talk to senior executives in companies. We normally end up talking about the technicalities of testing or even worse talking down to them as if they do not understand how really important testing is. The most important rule when talking to business people about testing is…

"Do not talk to them about testing"

You should instead try to make them feel comfortable in the knowledge that you as the person assigned the role of ensuring testing is done, have it covered.

Instead focus on how the testing approach you use is aligned to the business strategy of the company and how your role help the business be successful. Talk to them in terms of business and how you align testing to the business. What value does what you do add or prevent the business from losing money or customers?

Business people are focused on risk and trying to avoid risk that impacts the bottom line. When talking to executives instead of saying we should not be automating everything; talk instead about the risks associated with attempting to automate everything. The business costs to maintain or the risk of not uncovering information that could cause loss in value to the business. Talking about checking and testing can be useful to help people understand the value of what we do. It is important to present a balanced view and explain the benefit of both and how using both can mitigate risk.

Look at providing examples to the executives of the bad things that we are going to do to the customer if we do not test properly. Ask them how you can help with their decisions to help your clients and protect business value.

If your company is listed on the stock market, do you read or watch your company financial statements? This gives useful insights to their important values. Learning the financial language of your company can be useful when talking. With this information you can now tailor your discussion around the value your testing provides to the whole organization.

Many tester focus on the value that they provide or that the testing provides, instead focus on defining the value of the whole team delivering the product. Explain how the testing is aligned to delivering as a team rather than focusing on testing. Look at the company annual report; it highlights their risks and issues. Understand what that is and align your testing to this.

Most importantly, be prepared. If you know you are going to talk to these people understanding what is important to them, what motivates and drives them. Having this information can help build a relationship around their aspirations and make them feel you understand them and what their needs are.

As Keith states "Focus on the big stuff and work back from there." that is what the executives are really interested in.

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.

Counterintuitive Lessons from Test Consulting

- by Sunjeet Khokhar

The Samurai vs The Spice Grinder mind-set

"There are these two young fish swimming along and they happen to meet an older fish swimming the other way, who nods at them and says "Morning, how's the water?" The two young fish swim on a bit and then one of them looks over at the other and says "What the hell is water?" "

Point of the story being,

"The most obvious, important realities are often the ones that are hardest to see and talk about"

~ David Foster Wallace, 2005 - https://www.youtube.com/watch?v=8CrOL-ydFMI

I am pleased and proud to announce that what follows in these upcoming paragraphs is a smattering of clichés, but as with all clichés they are terribly easy to ignore, not reflected upon and questioned in the contemporary I.T. industry, as often they should be, because, well, they are clichés and everyone **should** just know how to do/not do them.

But do we?

Wallace's talk makes so much sense to me, i.e. in the hustle of achieving outcomes and shipping better software for our clients and customers, we ignore the **obvious** things in work and personal life.

We take a lot of pride in our ability to get lots done, in our ability to be busy and go **the extra mile** for our clients and customers. We fear that we are never doing enough to support our team and keep them motivated and fulfilled. We dread the perceptions that will be created if we are risk averse (**move fast and break things**), we wonder what will happen to our careers if we say "No" (** smart and get things done**).

We take pride and contentment in being busy in our work lives, but **why aren't we equally proud of being unbusy?**

Till couple of years ago, I was worshipping busyness with similar aplomb. I was of the opinion that I was at the **peak** of my professional prowess, I was leading a team of exceptional Testers, the majority of whom I had recruited and mentored. I was fortunate to have their respect and of my stakeholders. I had won several accolades in my company. I was grounded but full of professional pride. I had a deep personal bond with my team, I had invested a lot in their wellness but (can say now with the silver bulleted benefits of hindsight) not enough into my own.

I was only seeking external validation, recognition and fulfilment. As a result, I was ignoring the need to have unbridled play-time with my kids (on the weeknight of a big release), I was ignoring to ignore work emails while travelling to and back from work (I could not even wait to reach home to check work emails), I was ignoring not to remote in and reset the flaky automated test rigs during a long weekend break, I was talking attacks on my Team as attack on me,

And to make matters worse, I was not mindful of the impact some of my behavior was having on my team and my family.

My argument was, well if I **slow down** now, now when I am getting all this done, my career will perhaps be stymied.

In short, I was a **functioning work-alcoholic**.

You know what followed? Would not be hard to guess, aptly described by this song,

https://www.youtube.com/watch?v=maTo3QhSkyA

And captured by this picture….

(Image credit: YouTube)

I am not ashamed now to share that I crashed and burnt and stuttered into therapy.

I am also not ashamed now to share that I am a recovering work-alcoholic, and one of the (surprising) contributors in the meandering turn around has been the time that I have spent as a Test consultant.

Let me elaborate,

Image source: http://www.history.com/topics/samurai-and-bushido/pictures/samurai-and-bushido/samurai-brandishing-sword-by-felice-beato

At the height of my self-proclaimed **job satisfaction** peak, I was operating more like this samurai.

Moving at speed with precision and effectiveness. Wielding my sword skillfully, with great alacrity and confident judgement. Framing and solving problems that **mattered** to the business, fast. I was investing a lot in my Team and getting a lot of (quick and unexpectedly pleasant) outcomes. I was running brave experiments with my team and it was paying off.

The outcomes were the high that drove my work-alcoholism and the more fruits I reaped, the further intoxicated I got. I was totally outcome driven, a samurai assassin propelled by the number of trophies that I could gather.

But, obviously that did not **scale**.

Soon, I found myself in a job with a heavily bureaucratic environment, unsupportive bosses and apathetic colleagues. I applied my samurai mind-set to these environments and failed miserably. I had just changed from being a jockey atop a very nimble horse to being a "mahout" atop a very stubborn elephant and was expecting similar agility of outcomes.

This stunned me, broke me, my motivation dwindled, my drive stalled, my will to function in that environment had withered to a whimper.

I was not busy anymore, I was not getting outcomes at the pace I was accustomed to, I was not growing my team, in fact, and I did not have a team to grow anymore. I was not drawing on my core skills. Things were **too slow** for me there.

After being through therapy for several months, after leaving that job and after having been a Tester in product organizations for more than a decade, I stumbled upon Test consulting.

In the past couple of years I have consulted multiple businesses, as a Test Lead and a Senior Consultant, through Assurity consulting (www.Assurity.co.nz) in the adorable city of Christchurch, New Zealand.

I have been exposed a lot to business problems that has enabled me to inculcate the below mind-set to support my Samurai mind set. Let's call it, The Grinder mind-set.

The process you see in adjacent picture (https://www.youtube.com/watch?v=S2Olfr3PwrY) is old Indian practice of hand grinding spices to a paste, to be used in cooking curries. In today's age of many faster, more efficient and affordable alternatives available, a lot of household still carry this slow laborious process of grinding the spices not only because the taste of the curry is unmatched but also that it brings the cooks (mostly Indian house wives) together for a tete-a-tete and communal gossip.

Having grown up watching my mother, my aunts and their friends do it, I was in awe not only at the delicacies that they dished out **(the outcomes)** but also at the enjoyment that they derived from the seemingly monotonous manual grinding **(the process)**. Sometimes I wondered whether they were primarily there for the tete-a-tete and the heavenly curries were just an accidental outcome. The process of grinding mattered much more to the cooks than the curry.

Similarly, consulting has shepherded me into situations where I have been fortunate to learn and practice this contrasting mind-set required to **grind it out**, and not only endure it but to start reveling in it.

- Consulting has taught me that **business transformation and influencing change is a slow insulin drip rather than a searing burst of adrenalin**. Persistence and patience are paramount to building resilience.
- Consulting has taught me that to build resilience, **you need to fail and failure is practiced by doing what scares you the most** (Note - Coincidently, the "mantra" to success is the same)
- Consulting has taught me that **expecting relatively quick results (even though you put in your sincerest and most skillful effort) is a strategy bound to doom** in the long run (http://blog.samaltman.com/the-days-are-long-but-the-decades-are-short). Note - My definition of an atomic unit of "quick" is 1 year and of "long-run" is 3.

- Consulting has taught me that the **focus should be on getting yourself and your client invested in the process of reaching at an outcome, rather than just chasing outcomes**. Validate your problem solving process over just validating the outcome.
- Consulting has taught me that **influencing myself (and my behaviors) comes way before trying to influence others**
- Consulting has taught me to ask this question daily, ****what is that one thing that I learnt today, that I did not knew yesterday?** **
- And consulting has taught me that **not being mindful enough to ask this question and/or not taking an action to act, when the answer is no, is letting down your craft**.

Just to clarify, I'm not going ga-ga over consulting's magical healing powers, these are lessons which can be learnt in any job from product development to delivering mail in remote Himalayan countryside. Unfortunately these are learnings that I did not gain (because I was just a Samurai) but am thankful that being a consultant has provided me with opportunities to draw these out and apply in my daily professional life.

Summary -

Please don't take me wrong, I am not suggesting that you stop looking at outcomes or be goal driven. Neither I am lecturing you on how to manage stress in your work life nor that you form navel gazing philosophical view of work and stop shipping great software.

Rather I am articulating that **the Samurai (Outcome focused) and Grinder (Process focused) mind-set are the Yin and Yan of being a whole professional craftsperson.**

Be fearless, bring your Samurai out,

Be skilled at the usage of Test tools, be an excellent Exploratory tester, be an astute coder who can write scalable test frameworks, Be the person who gets stuff done

But also relish the process and champion the grind,

Practice and champion mindfulness, Self-reflect daily, don't leave culture to anyone (build your own), let someone go to uphold your company values, create scalable/replicable/lean problem solving frameworks, document & share knowledge, coach selflessly

Be a great swimmer, but also know what water is.

Having started his Testing career in 2004, with testing C++ APIs for desktop licensing applications, Sunjeet has tested web applications for Doctors, mission critical systems for Fire fighters, ERP systems for Builders, .NET based applications for providing a better casino gaming experience for Grandmas. He has worked with an equal passion (but varying success) across Product development organisations, Consultancies and public sector.

Sunjeet is grateful to have had opportunities to serve as a Test coach, people leader, technical mentor and now a Test consultant with Assurity Consulting in Christchurch, New Zealand.

He is driven by the success of people around him, is a keen student of organisational behavior and firmly believes that we can be better craftsmen by being better humans first. He blogs about Testing, Leadership and poetry at https://thereluctanttester.wordpress.com/

Love to Write?

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

Send your articles to editor@teatimewithtesters.com

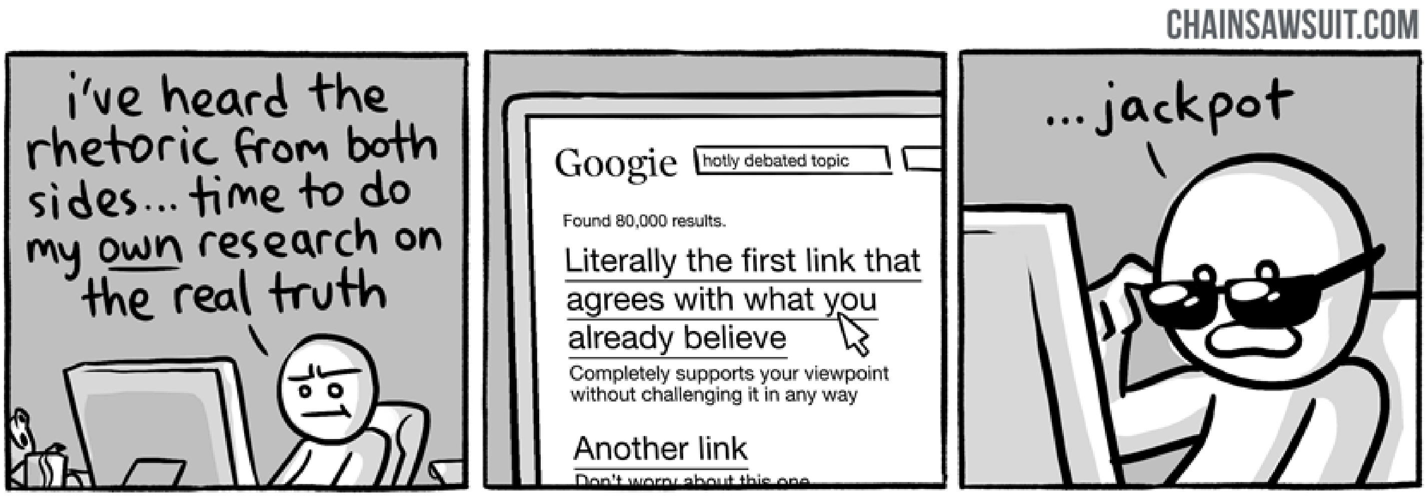
In the school of Testing
for your better learning & sharing experience

Mapping Biases to Testing - part 3!

- by Maaike Brinkhof

I use terminology from earlier blog posts about biases. If you have missed those posts, read part 1 here. I explain the terminology there. In the second post I wrote about the Anchoring Effect.

Let me state the 'bad news' up front: you cannot fully avoid the confirmation bias. That's actually a good thing, because if you could you wouldn't be human. We jump to conclusions (with System 1) when our assumptions are likely to be correct [1] and a mistake is acceptable. For example, when you meet a new person you immediately judge him or her based on stereotypes, what type of clothes the other person wears, his/her posture, etcetera. This happens so quickly that you cannot stop it.


CHAINSAWSUIT.COM
i've heard the rhetoric from both sides... time to do my own research on the real truth
Google
hotly debated topic
Found 80,000 results.
Literally the first link that agrees with what you already believe
Completely supports your viewpoint without challenging it in any way
Another link
Don't worry about this one
...jackpot

However, jumping to conclusions can be a bad thing when the situation is unfamiliar, the stakes are high and when there is no time to collect more information [2] . This screams 'testing!!' to me. We are often dealing with unfamiliar situations, the stakes are high and we usually face a deadline. How do we deal with this? Let's explore what the confirmation bias has to do with testing.
The confirmation bias is an umbrella term with more specific biases in its category: the halo effect, 'What you see is all there is' (WYSIATI) and the availability heuristic, to name a few from "Thinking, Fast and Slow". We'll also see that heuristics, which are very important in our work, are strongly linked to the confirmation bias.

The Halo Effect

The halo effect is explained as "the tendency to like or dislike everything about a person". I think this can also apply to systems or new parts of a system you are testing the first time. A first impression is hard to erase. Confession time: after a while in the same team I started expecting a certain level of quality from the different developers. Developer A was a very structured and organised person, taking pride in developing clean code. He would involve me during development and ask questions. When it was time to do an exploratory test session (after the automated tests were written by us together) I was liking everything about the software. I was already biased in a positive sense.

Developer B was a sloppier person; not writing unit tests, not even starting up the application locally before 'handing it over to test'. Without discussing how not Agile this approach is, at that point I was already expecting his work to be crappy. Even when, over time, developer B was coached by other developers and improved his way of working, I still tested his code with more suspicion. I wonder how many bugs I missed in developer A's work, because of the halo effect.

'What you see is all there is' (WYSIATI)

This one has really scary implications for testing. WYSIATI is "an essential design feature of the associative machine. It represents only activated ideas. Information that is not retrieved from memory might as well not exist." I basically read this as: you cannot test what you do not think about. Sounds completely logical to me, but this is often not how testing is sold! We are casually using terms like '100% coverage', 'completely tested' and the like. We might test a product extensively and thoroughly, but that doesn't mean we catch all the errors. That's personally the part of testing I have always struggled with. There's a trade-off between time and risk. You have to make the leap of faith towards going live even though you know that scientifically there is always a chance (or is it a certainty?) that a (horrible) bug might be lurking around the corner.

Availability Heuristic

This heuristic is described as "a mental shortcut that relies on immediate examples that come to a given person's mind when evaluating a specific topic, concept, method or decision." It is about how easy it is to think of examples. The easier that is, the more you believe what you think it truthful. You can convince yourself of a 'truth' simply because you cannot think of evidence that proves your hypothesis wrong. To me, the availability heuristic comes into play when you decide whether you have tested enough. It happens when you have been testing an application or part of an application for a while and you are not finding interesting things anymore; you are not learning anything new about it anymore. You might decide: I have tested enough; I am confident I have found all the bugs. You cannot think of other tests or test techniques at that time. Put in the context of the availability heuristic: it is no longer easy to think of examples (test ideas) so you are convinced you have done enough.

Again and again, I have found this to be dangerous territory. It has happened to me that I was confident about the quality of a new part of an application, only to come back into the office the next day (with new energy and new ideas) to find an absolutely terrifying bug. A bug that, if I were to port myself back to the day before, I didn't think would exist, simply because that example (specific test idea) didn't come to mind easily. To avoid the availability heuristic, you need the power of collaboration.

I want to make an extra mention regarding other types of heuristics. I love heuristics a lot, but there is a danger hidden in them. The more experienced you become, the more you might start to rely on heuristics. It gets easier to recognise patterns in certain situations. Your associative memory is triggered a lot when you are an experienced tester. Just be aware that this associative memory is strongly linked to the confirmation bias. Stay open minded. Stay curious for other people's view on things. That can help you avoid the confirmation bias to a certain extent.

Conclusion

System 2 is in charge of doubting and unbelieving [3], which is very important in testing! So when the stakes are high, try to involve System 2 thinking. Be kind to yourself though, it simply is not possible to always involve System 2. You would get very tired. Just realise you are vulnerable to different manifestations of the confirmation bias. Take a step back once in a while and ask someone else to pair up with you.

PS: have you read "Thinking, Fast and Slow" yet? No? Run to the (internet) bookstore, off with ya! See you again!

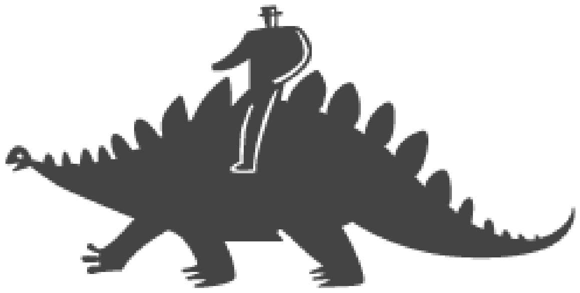
[1] Page 79, Thinking, Fast and Slow (Kahneman)
[2] Page 79, Thinking, Fast and Slow (Kahneman)
[3] Page 80, Thinking, Fast and Slow (Kahneman)

Back To Index

Maaike Brinkhof - Test Coach at ING, 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.



Outdated Testing Concepts #1

- by Viktor Slavchev

End of the previous year and the beginning of the current one are normally the times we are determined to make **new year's resolutions**, which most of the times turn to be **big, fat lies**, but anyway. I am not writing this to make resolutions, but to **set goals**. And we as a testing community have a lot of work to do in order to progress, since our craft was, as James Bach called it, "in perpetual childhood" for more than couple of decades. I believe the reason for this is there's still a lot of misconceptions, myths or just **outdated beliefs** about testing that **need to be declined**, so we could progress in the right direction and leave the useless luggage of principles that never worked or were ineffective.

So here is our list:

Outdated concept # 1: Testing is easy. Everyone can do it.

This concept was around for far too long and I believe every one of us had to **explain at least thousands of times how our job isn't simply "play with the product" or "break something" or "find the bugs" in it**. Testing craft is profound and complex as it possibly gets, and no, testers are not failed programmers, they just don't mix with programmers. Anyway, this is useless info, since many readers of this are testers, this isn't helpful at all.

The thing that really surprises me is that c**ompanies accepted that concept for granted and even adapted their processes to it**. All the outsourcing companies try to make new testers believe that it's isn't up to tester's specific skills and knowledge of the product, nor the experience he has, but the documentation of the processes itself. They will make you believe that testing could be performed by mindless morons and you **should write your test plans and test cases as if they going to be read**

by mindless morons. This way, they assure once you are gone, next totally green and incompetent tester will take your spot doing **just the same**.

Their biggest fear is when you try to **defend your value** by explaining how complex a testing process is and that in fact **IS** scientific experiment with all it' assets, therefore **needs educated expert in order to be performed** in the right way and **effectively**.

I had an interview once, the interviewer was only concerned if I could write test cases, just that. So, we had an argument, that testing **shouldn't** be determined in a specific set of actions to be effective, that this is **only one approach** to do it, that testing **documentation is not test cases** in the best way and that testing process is far too complex and fluid to define it as a set of rules, it depends on **observation**, **critical thinking**, **oracles**, **assumption**, **risk assessment**, **heuristics** etc., … Poor guy was almost shocked. It was a **cataclysm in his poor narrow view of what testing is**, finally he got relieved by knowing I am able to write test cases, but I don't like to, because it's boring and doesn't document the way that **I perform testing**. I finished with the statement that this is so-called "best practice" we tend to stay away from, because **it's not always relevant**. The answer was: "Well, you see, these are best practices for a reason".

So, **how can we react** against this and justify the importance of testing as an experimental process and not just scripted activity?

- In first place you don't have to be afraid to **stand for your beliefs**, make sure you are able to explain the work you do, not just do it.
- On the opposite, make sure you could give clear examples "why" and actually apply what you believe in and not just brag about it. **Demonstrated knowledge is one of the most valuable skills in any craft.**
- **Don't expect** anyone to know the specifics of testing, it is our job, but give the details carefully and in understandable manner for a person that has no background in software testing, as they could cause **a lot of confusion** if served at once.
- Knowing your craft and being able to answer the question "why" is vital for your development as a tester. As you see from the opponent's view-point, in my interview, the answer to the question "Why follow best practices?" varies from "Because we have to", to "because everyone else does". As you can see, **it's not an enormous effort to question such an explanation** and we could definitely **do better than that**.

I wasn't expecting this to get so long, but as it gets in testing, things are dynamic and we should adapt to them.

Important notice: Some important sources that provoked me to get to this point are the book **"Lessons learned in software testing" by Kaner, Bach and Pettichord** as well as the videos from **Introduction to BBST by Cem Kaner.**

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

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

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

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

I feel that the Software Quality Assurance (SQA) is expected to ensure that the software conforms to quality standards defined and is a continuous process that accompanies the entire life cycle of the product.

The most relevant quality measures of the software according to me can be grouped as follows:

Does product do what it was designed to do– Functional requirement
No unexpected (bad) behavior – Functional requirement
Performance – Non-functional requirement
Safety -Non-functional requirement

The software tester is one who creates the testing strategy for the team and thus for the company they work for.

The capabilities required to assure quality are various, such as maintaining governance during test development, collaborating with other professionals, knowing how to communicate, being able to move in changing scenarios that present increasingly different functionalities and by always keeping a clear objective to ensure product quality.

We conclude that the software tester is the architect of the testing process.

For me, the first step which helps define the entire testing process is the definition of the **User Acceptance Criteria (UAC)** that establishes the proper operation and completion of the product. More specifically, the UAC is a set of instructions, which specify the functional requirements and not of the product, by defining the criteria of quality for the same. The UAC constitute our "Definition of Done" in the form of an Epic, Feature and Story.

The definition of UAC allows the team to figure out when it's done and not to forget important borderline issues.

The UACs are produced through collaboration between developers, testers, and product owner and are created before the development, during the planning phase. UACs are expressed at high levels (conceptual, not detailed) and then in the form that works best for the team (keep it minimal).

During Kick off, QA and developer, in a 10-15 minute session must explicate the **User Acceptance Test (UAT)** in order to define the behavior of the system and ensure the realization of the product as expected.

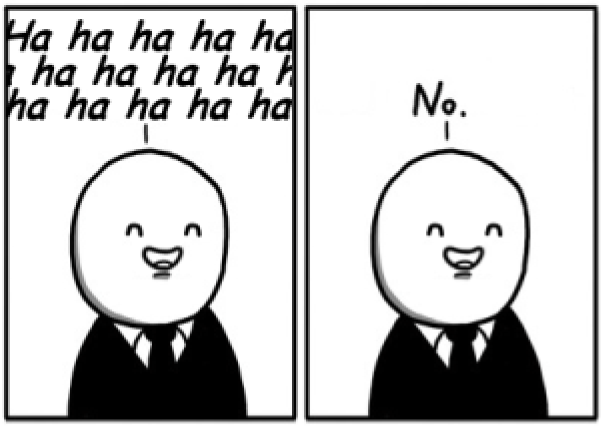
The QA and developers agree, also, the tests to implement, sometimes also including the product owner and the other stakeholder. The implementation of the test takes place during development (ideally test-first).

It is considered appropriate to build a **Map of User Acceptance Test automatic and/or manual** in order to improve the testing and development process. They will be designed at the level of User Tests, Unit Tests, Integration Tests, Alpha/Beta Tests, Exploratory Tests, Performance Tests, Security Tests, and Static Tests.

So that QA does not become a bottle neck the quality should be the responsibility an entire team.

Why "Test Tower" instead of "Test Pyramid"?

Because the pyramids are the XXVII century A.C.

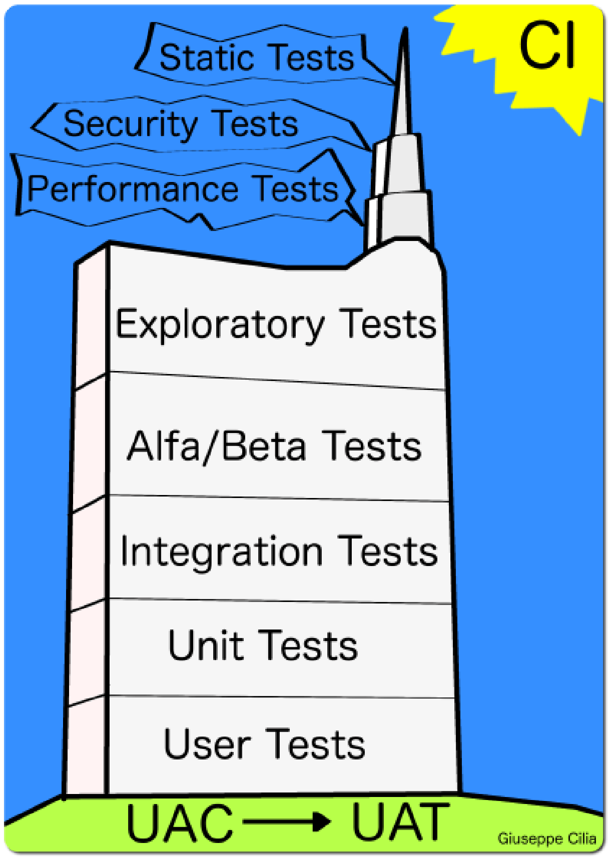


The test pyramid is a concept developed by Mike Cohn, described in His Book "Succeeding with Agile".

The pyramids of testing allow you to see which test levels to be performed and highlight the amount of tests for each type. Over the years we have developed various types of testing pyramids, but now we wonder if we still need to see the testing through a pyramid.

As well as products developed by companies that are revolutionizing the behaviors and habits of people (social networks, search engines, online reservations, online shopper, car-bike sharing, online services) go towards quality and focus on it to differ from continuously rising competitors. I feel even the testing has to go in that direction. Therefore I propose to **speak of towers, towers where testing is not referred in terms of quantities for each level but the quality of each of those tests (types) for each level!**

Instead of displaying the levels and types of tests through the pyramids I prefer to take inspiration from the palaces and towers (Unicredit Tower), populating the city where I live for a while, Milan.



As mentioned in the first part of this article, it is essential to clearly define the User Acceptance Criteria for PO, UX, Team Manager, and convert them to User Acceptance Tests. It is essential also, to do the following:

- Define a set of User Tests with the UX department, before and after the development of the Feature.
- Create Unit Tests and Integration Tests, backend and frontend and from backend to frontend (End To End Tests).
- Create user groups who want to try out the product with Feature realized, internally and/or externally of the company.
- Maximize the Exploratory Tests but remember that the software is full of surprises. No matter how careful or skilled you are, when you create software it can behave differently than you intended. Exploratory testing mitigates those risks.
- Remember that Performance, Security and Static Tests will highlight other critical aspects of the produced Feature, up to rethink it completely.

  Finally, the CI (Continuous Integration) allows the control of the Feature to every Build.

Finally, although it is important to define the amount of tests to be performed for each test level (Test Pyramid), the quality of each single test for each level of testing and careful planning of the testing process and strategy will lead to Products high quality (Test Tower).

Giuseppe Cilia, is an Italian quality assurance engineer of software with a continuously growing expertise gained from self-teaching and research on top of a solid academic foundation. He does his best to meet the business goals by implementing the best practices of test design (automatic and manual). He also understands the importance of user experience when it comes to mobile applications and he tries to keep in mind, the user during the entire testing process.

Giuseppe is very active on the forums of testing and does not miss opportunities to attend conferences that hold his interest.

Giuseppe works in Catania, Sicily, and together with his boss he has learned many things about testing on web-browser. He is a full stack quality assurance engineer on web-browser, m-site and iOS/Android/Windows Phone applications.

He is passionate Agile tester and loves to read Tea-Time with Tester magazine for his continuous learning.

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

Episode 1

Talking Mobile App Testing

with Daniel Knott

TV for Testers

Your one stop shop for all software testing videos

Sharing is caring! Don't be selfish ☺

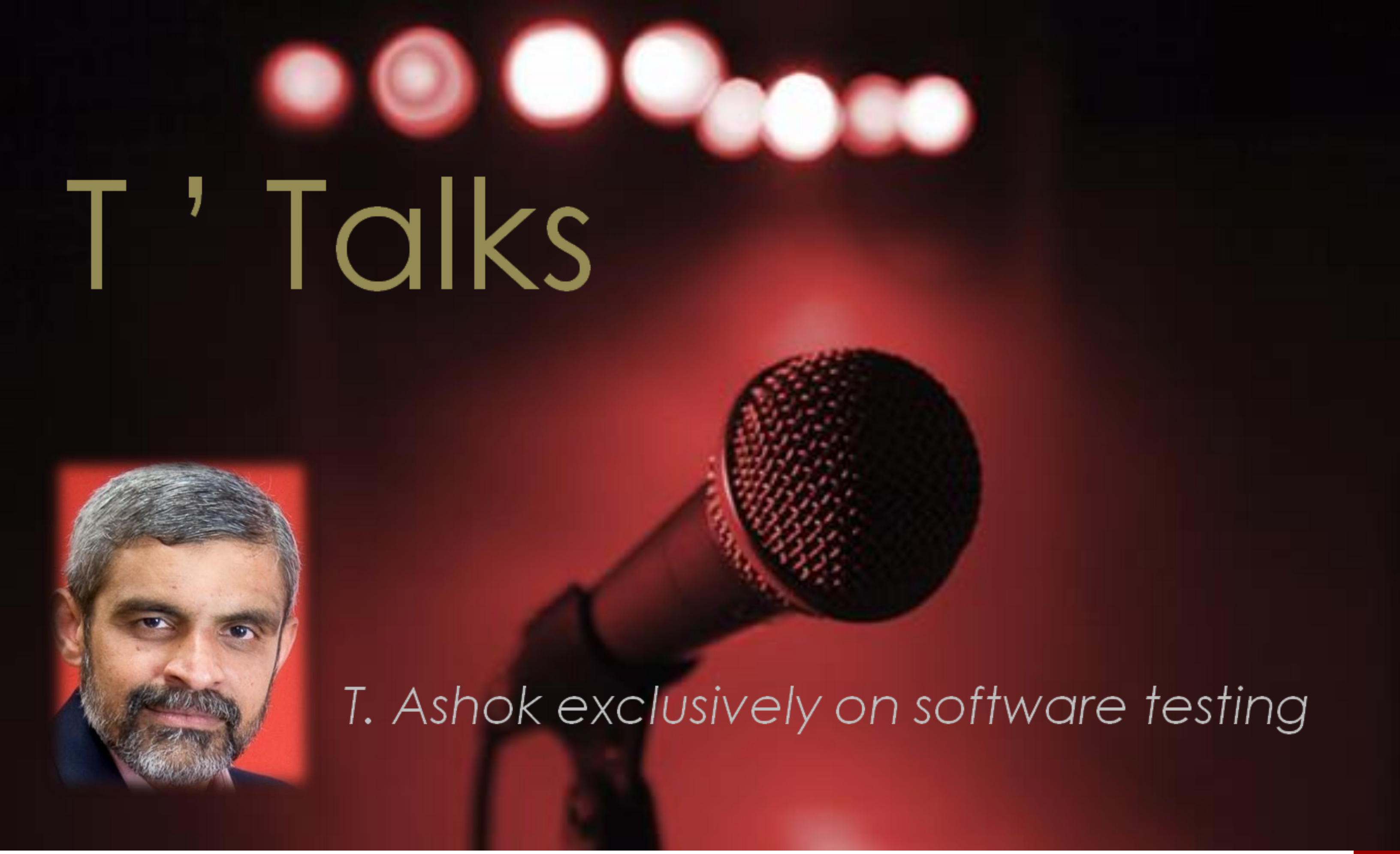
Share this issue with your friends and colleagues!

Happiness is....

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

Like our FACEBOOK page for more of such happiness

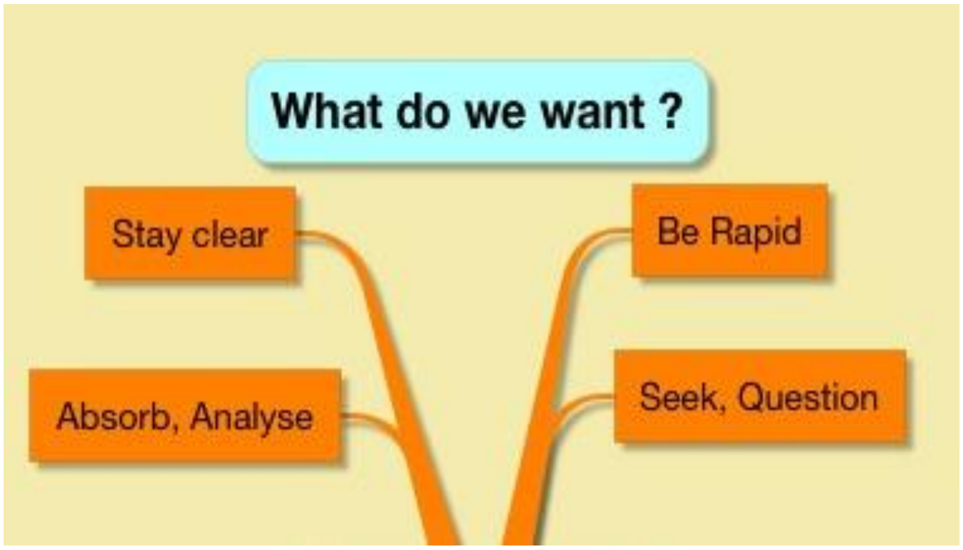
https://www.facebook.com/TtimewidTesters



The Elements of Good Testing

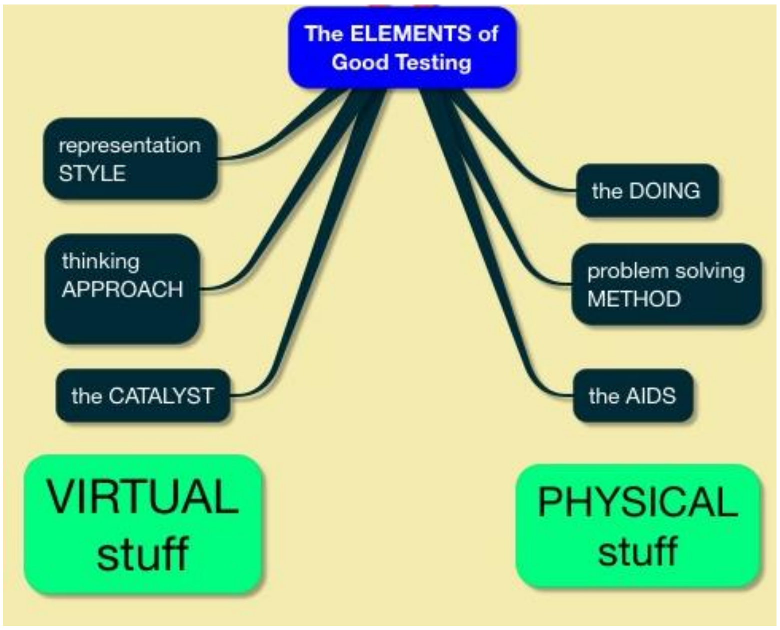
We want to test rapidly, deal with incomplete information yet stay clear, absorb vast amounts of information, process and analyse well, be a seeker and question deeply to test very well.

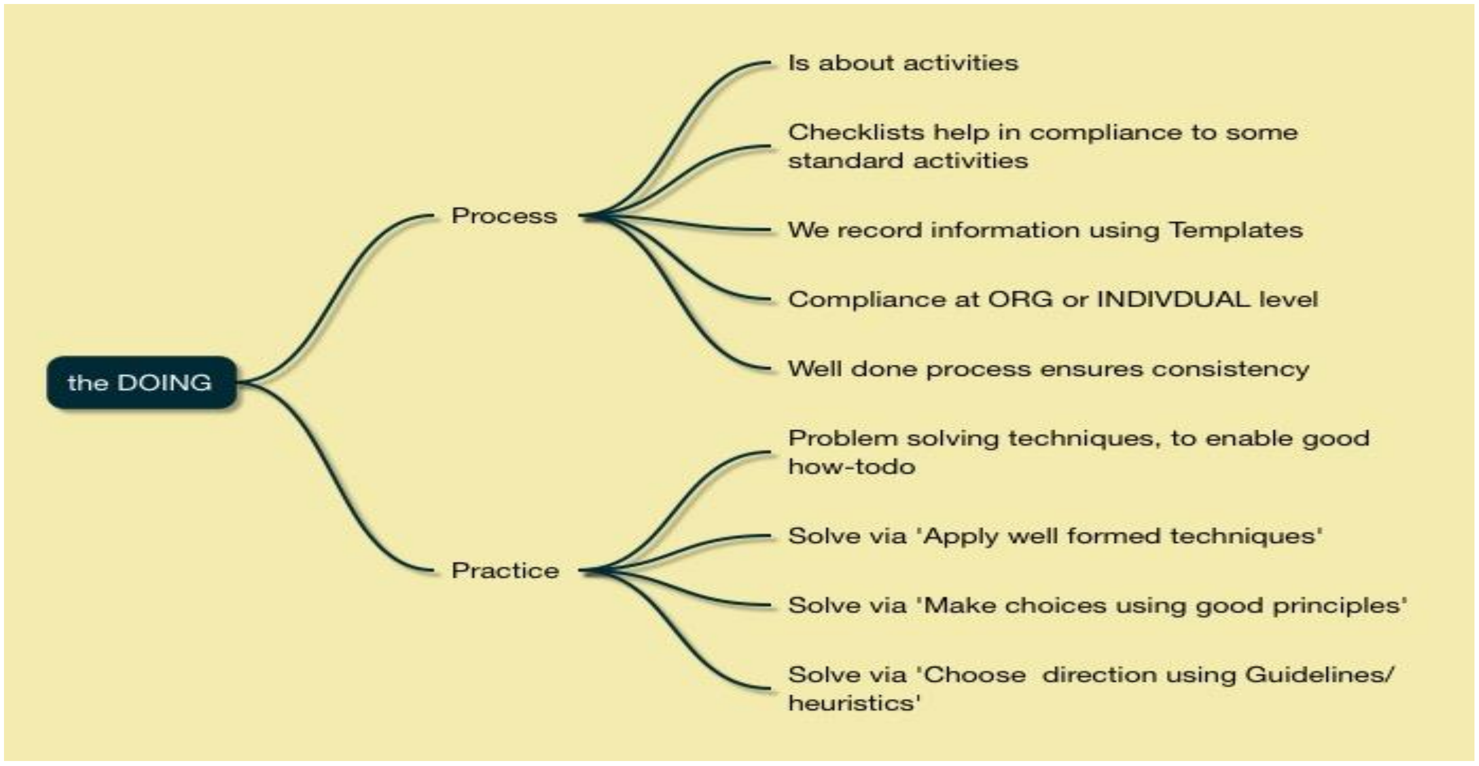
Good critical thinking skills are the order of the day.



So how do we accomplish this?

Hmmm, it seems like the elements to good testing may be: How we do (the DOING), how well we do (problem solving METHOD), the AIDS that we use to assist in our DOING. In addition to these physical stuff, it also is non-technical virtual stuff of how well we are engaged in the act to be in a flow, think creatively in addition to being deductive and logical and the role of language styles and diagrams/sketches in unleashing the right brain and creativity.

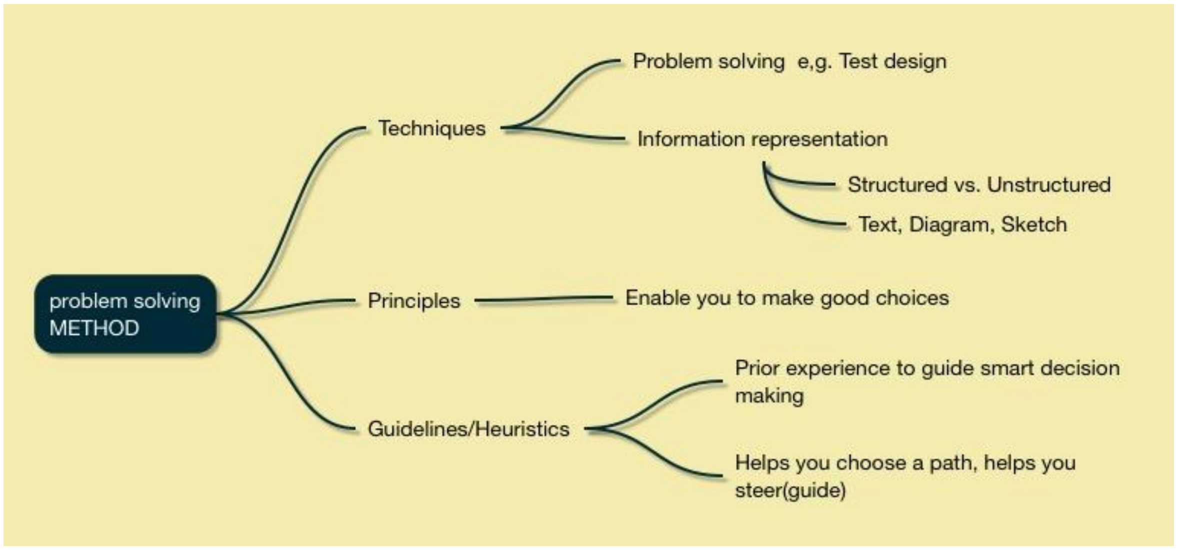



the DOING
Process
Is about activities
Checklists help in compliance to some standard activities
We record information using Templates
Compliance at ORG or INDIVDUAL level
Well done process ensures consistency
Practice
Problem solving techniques, to enable good how-todo
Solve via 'Apply well formed techniques'
Solve via 'Make choices using good principles'
Solve via 'Choose direction using Guidelines/ heuristics'

Let us look at "the DOING", as to how we perform the various activities.

Processes at a higher level list (org/division) list who, when , what needs to be done to ensure consistency of work by enabling clarity of what needs to be done. Templates provide a simple boiler plate for seeking, recording information aided by checklists for activities to enable a consistent way of doing. The key benefit of processes is largely being consistent, whilst the effectiveness depends on the individual's practice of doing - of choosing the right techniques , using it correctly, make intelligent choices by applying good principles, relying on ones/others experience(s), available as possible guidelines/heuristics to choosing the most appropriate path to doing. It is a winning combination of PROCESS & PRACTICE that ensures that outcomes of activities are consistent, effective and efficient. **It is "the DOING" element**.

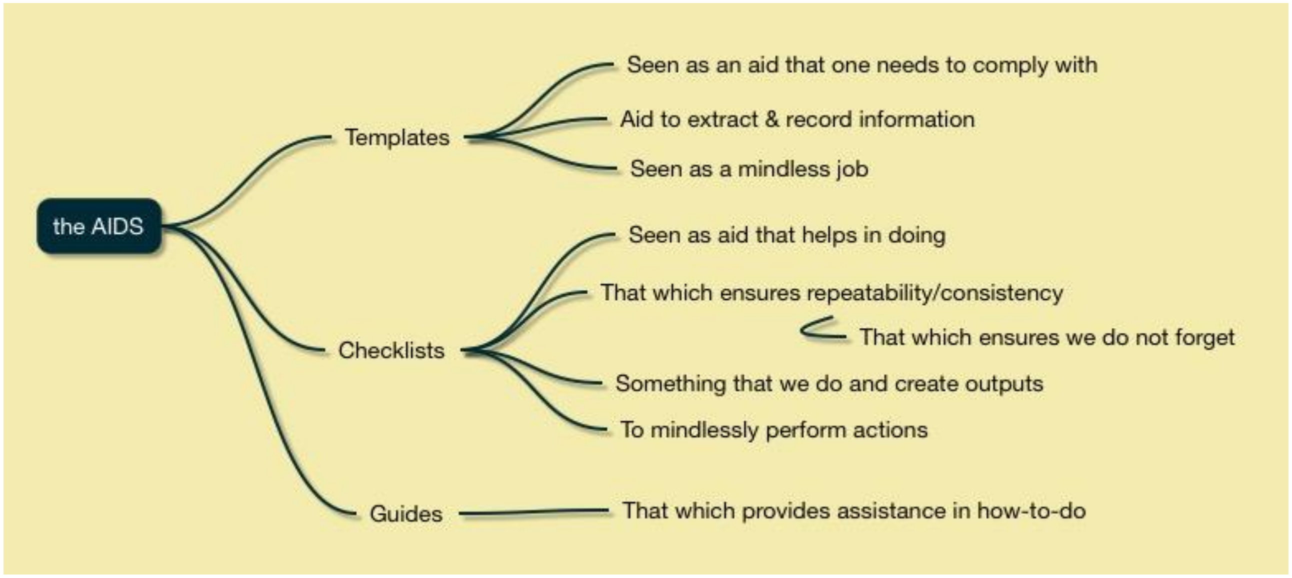
Not only is it necessary that we have clarity of "what-do-do", it is also necessary to doing it well. Be it any activity in the lifecycle like read/explore to understand or come up with scenarios/questions, it is the "how-


problem solving METHOD
Techniques
Problem solving e,g. Test design
Information representation
Structured vs. Unstructured
Text, Diagram, Sketch
Principles
Enable you to make good choices
Guidelines/Heuristics
Prior experience to guide smart decision making
Helps you choose a path, helps you steer(guide)

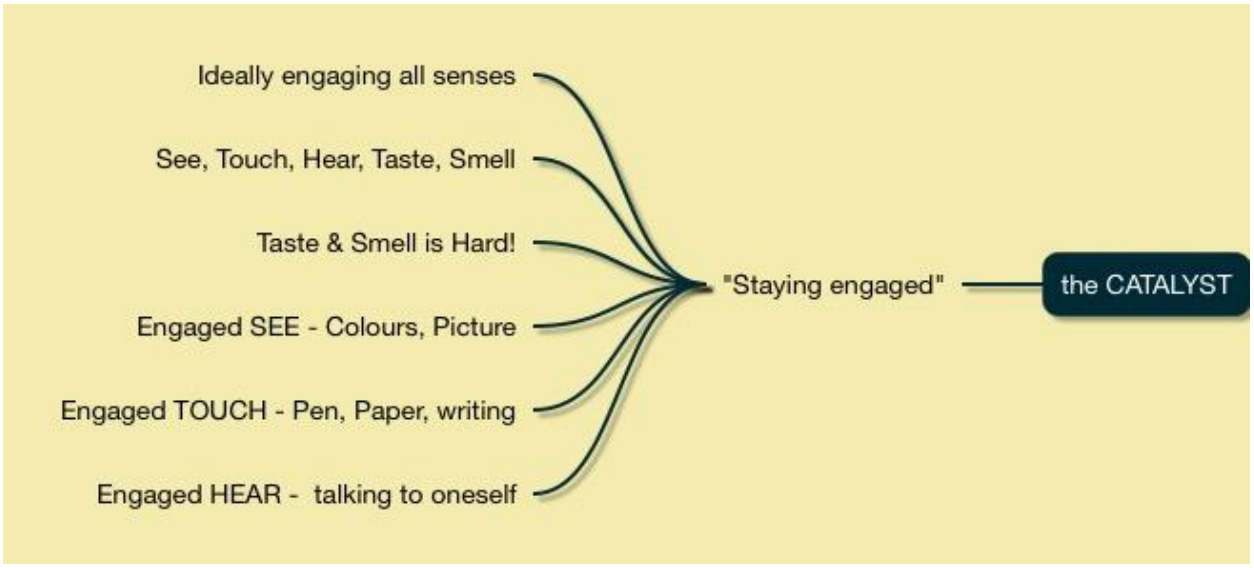
we-do" i.e. "**problem solving METHOD**" that matters for effective testing.

The methods of problem solving are sometimes perfectly clear, like using a technique that is person independent or those driven by experience codified as guidelines/heuristics or decision making principles to make choices. A good practice that is largely individual based is the backbone to effective implementation of the various activities that we talked about.

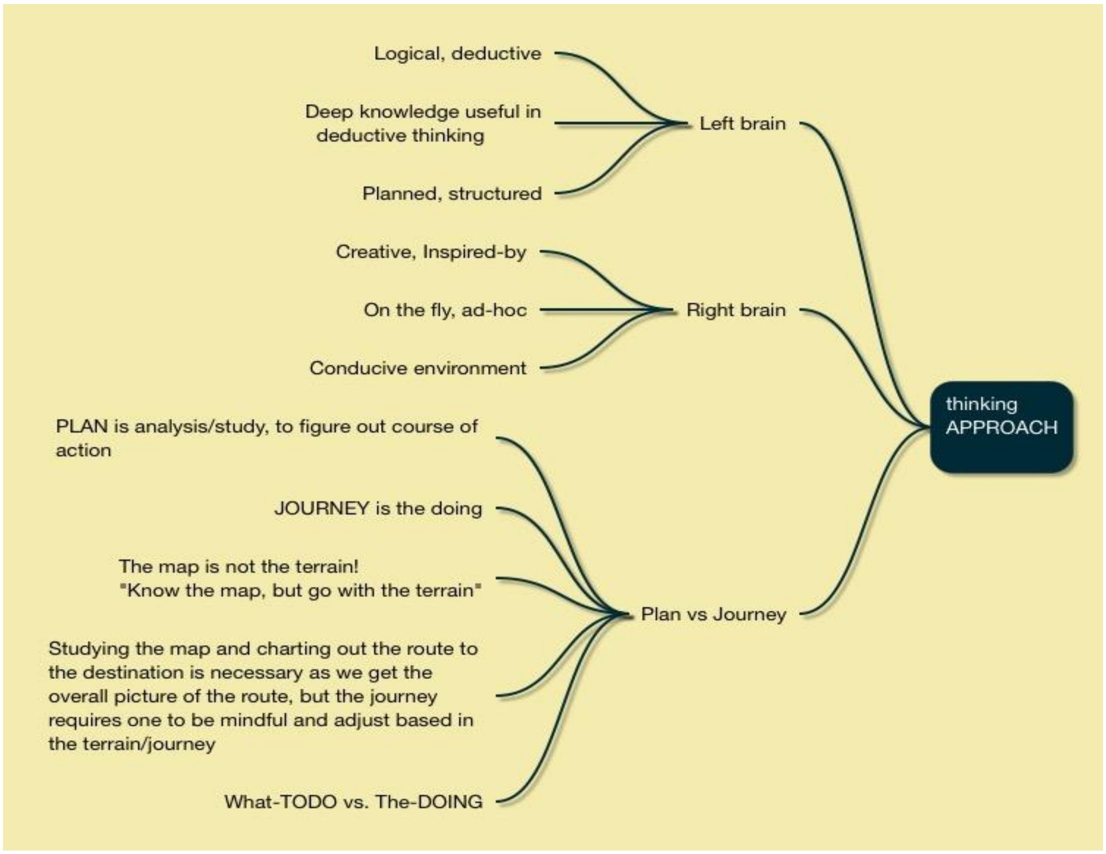
To ensure good implementation of activities or the method to solve problems in the in activity(ies) "**the AIDS**" like templates, checklists, guidelines are often used.


the AIDS
Templates
Seen as an aid that one needs to comply with
Aid to extract & record information
Seen as a mindless job
Checklists
Seen as aid that helps in doing
That which ensures repeatability/consistency
That which ensures we do not forget
Something that we do and create outputs
To mindlessly perform actions
Guides
That which provides assistance in how-to-do

Note that process aids like Templates, Checklists aid in compliance ensuring we do not miss any fine grained information or activity sometimes quelling the creative side of problem solving. Note in certain situations when it is required to be very diligent checklists are very useful. There is a very interesting article in the book "Differential diagnosis" on how checklists were very instrumental in cutting down infections in hospital ICUs enabling faster recovery and saving a whole lot of money. "the AIDS" need not only be drudgery assistants but can be also intelligent ones to extract of better information or enable intelligent processing. Beyond the DOING, the METHOD of doing and the AIDS lies interesting "human stuff" that as individuals we can exploit to do far better work. The first one is ENGAGEMENT; of engaging all the senses especially the "SEE/TOUCH/HEAR", immersing oneself and being in a flow, to unleashing the creative senses, being sharp and therefore doing far better work.


Ideally engaging all senses
See, Touch, Hear, Taste, Smell
Taste & Smell is Hard!
Engaged SEE - Colours, Picture
Engaged TOUCH - Pen, Paper, writing
Engaged HEAR - talking to oneself
"Staying engaged"
the CATALYST

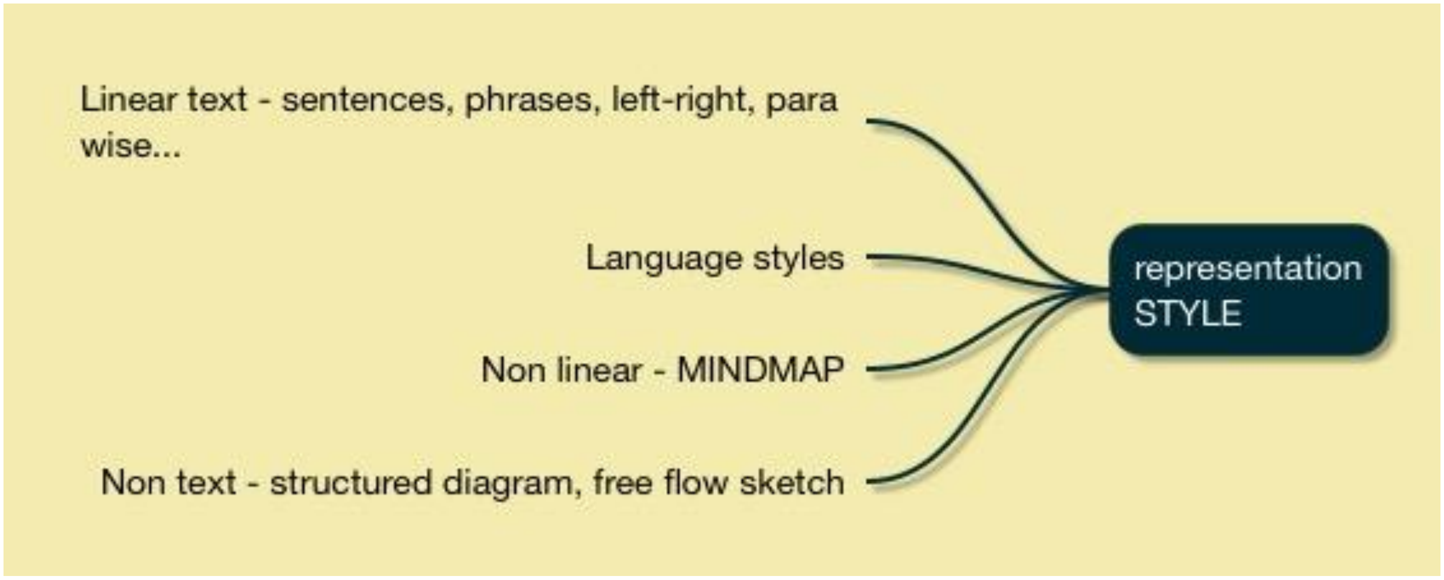
The CATALYST enables far better engagement of the senses to be able to unleash the sharp deductive/logical thought process powered by the left brain and the creative energy of the right brain.


Logical, deductive
Deep knowledge useful in deductive thinking
Planned, structured
Left brain
Creative, Inspired-by
On the fly, ad-hoc
Conducive environment
Right brain
PLAN is analysis/study, to figure out course of action
JOURNEY is the doing
The map is not the terrain! "Know the map, but go with the terrain"
Studying the map and charting out the route to the destination is necessary as we get the overall picture of the route, but the journey requires one to be mindful and adjust based in the terrain/journey
What-TODO vs. The-DOING
Plan vs Journey
thinking APPROACH

It is no more just about cold plain logic, but of thinking non-linearly, creatively, of enjoying the journey rather than just devise a detailed plan analytically. The "**thinking APPROACH"** of exploiting the combination of left and right brain paves the way for breakthrough thinking, of coming with variant approaches of not treading the same path, of staying curious.

The harmonious engagement of left and right brain, of being logical yet creative, be able to plan meticulously of how-to-do, yet course correct and adjust rapidly, requires appropriate stimuli which is where "**the representation STYLE"** comes in.

It is about stimulating the various senses via crisp text, of non-linear representation like mind map, using creative sketching/diagram, and finally the writing styles of being imperative, declarative, and interrogative to engage, sharpen, focus yet enable divergence.


Linear text - sentences, phrases, left-right, para wise...
Language styles
Non linear - MINDMAP
Non text - structured diagram, free flow sketch
representation STYLE

This article is not pedagogic, nor prescriptive, but an attempt to weave the various elements into a interesting mosaic that enables one to understand the beautiful interplay of **the DOING activities, the METHOD of doing, the AIDS, the CATALYST that aids doing, the thinking APPROACH, representation STYLE** 'jargonised' as process, practice, templates, checklists, guidelines, techniques, heuristics, principle, good practices, creative/ad-hoc, disciplined, mind map, sketch, deep knowledge, experience.

It is the unique combination of these based on the context that fosters effective testing yet be efficient and consistent.

Cheers. Have a great day.

Back To Index

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

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

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

OnlineTestConf

The Online Testing Conference

THE FIRST 100% ONLINE TESTING CONFERENCE!

November 29-30, 2016

Explore Testing Strategies, Automation and Technical Improvements with leading QA professional from around the world.

Sponsored by PractiTest

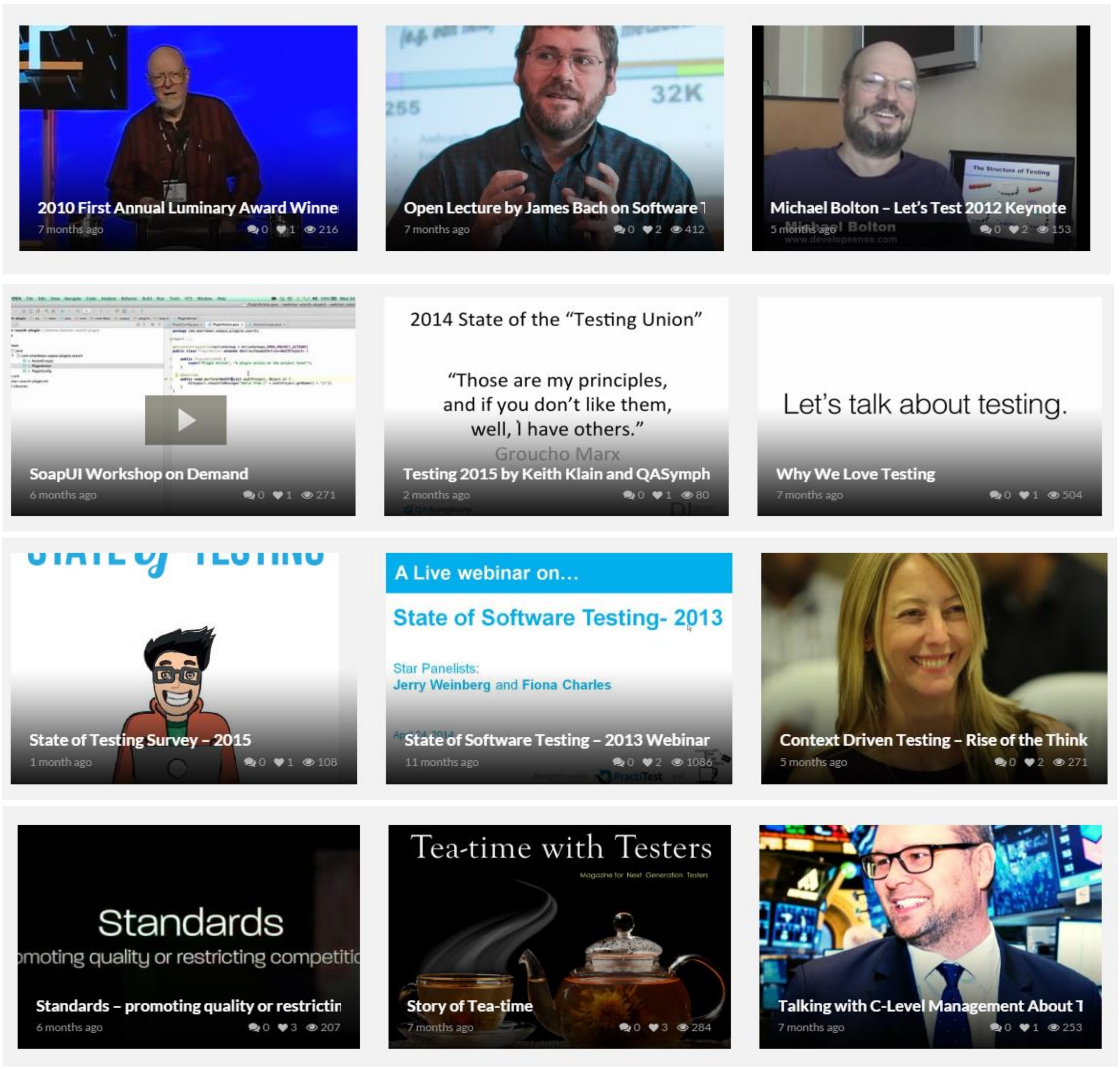
Save your seat: www.onlinetestconf.com

Attendance is free & Completely online.

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

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…

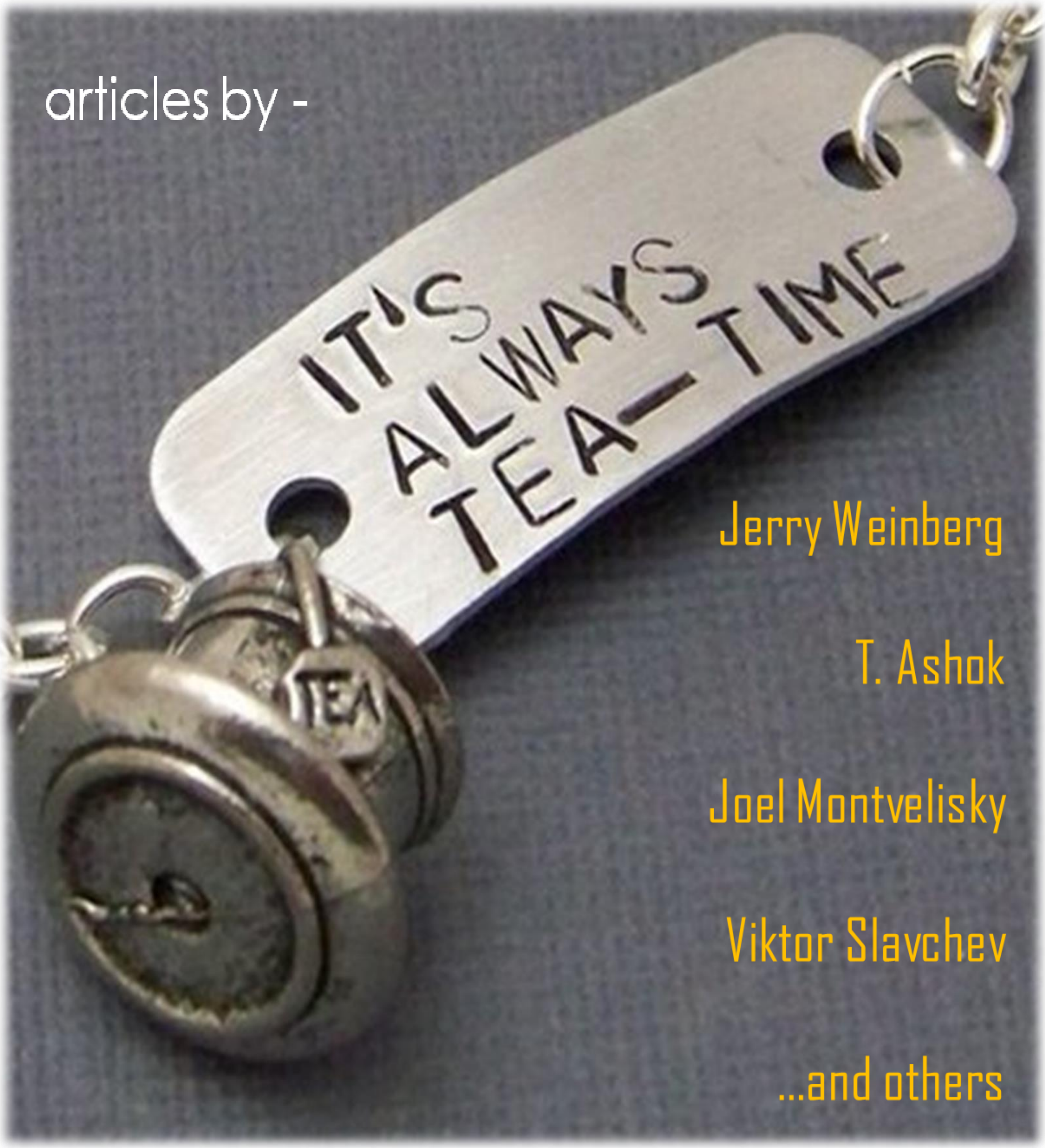
Now we have some more awesome offerings at pretty affordable prices.

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 nexxt issue

articles by -



Jerry Weinberg

T. Ashok

Joel Montvelisky

Viktor Slavchev

...and others

our family

Founder & Editor:

Lalitkumar Bhamare (Pune, India)

Pratikkumar Patel (Mumbai, India)

Lalitkumar

Pratikkumar

Contribution and Guidance:

Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)

Jerry

T Ashok

Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover page image – Metaphrasi

Core Team:

Dr.Meeta Prakash (Bangalore, India)

Dirk Meißner (Hamburg, Germany)

Dr. Meeta Prakash

Dirk Meißner

Online Collaboration:

Shweta Daiv (Pune, India)

Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)

Kiran Kumar

Chris

Romil

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

To get a **FREE** copy,

Subscribe to mailing list.

SUBSCRIBE

Join our community on

facebook

Follow us on - @TtimewidTesters

www.teatimewithtesters.com

Give Feedback