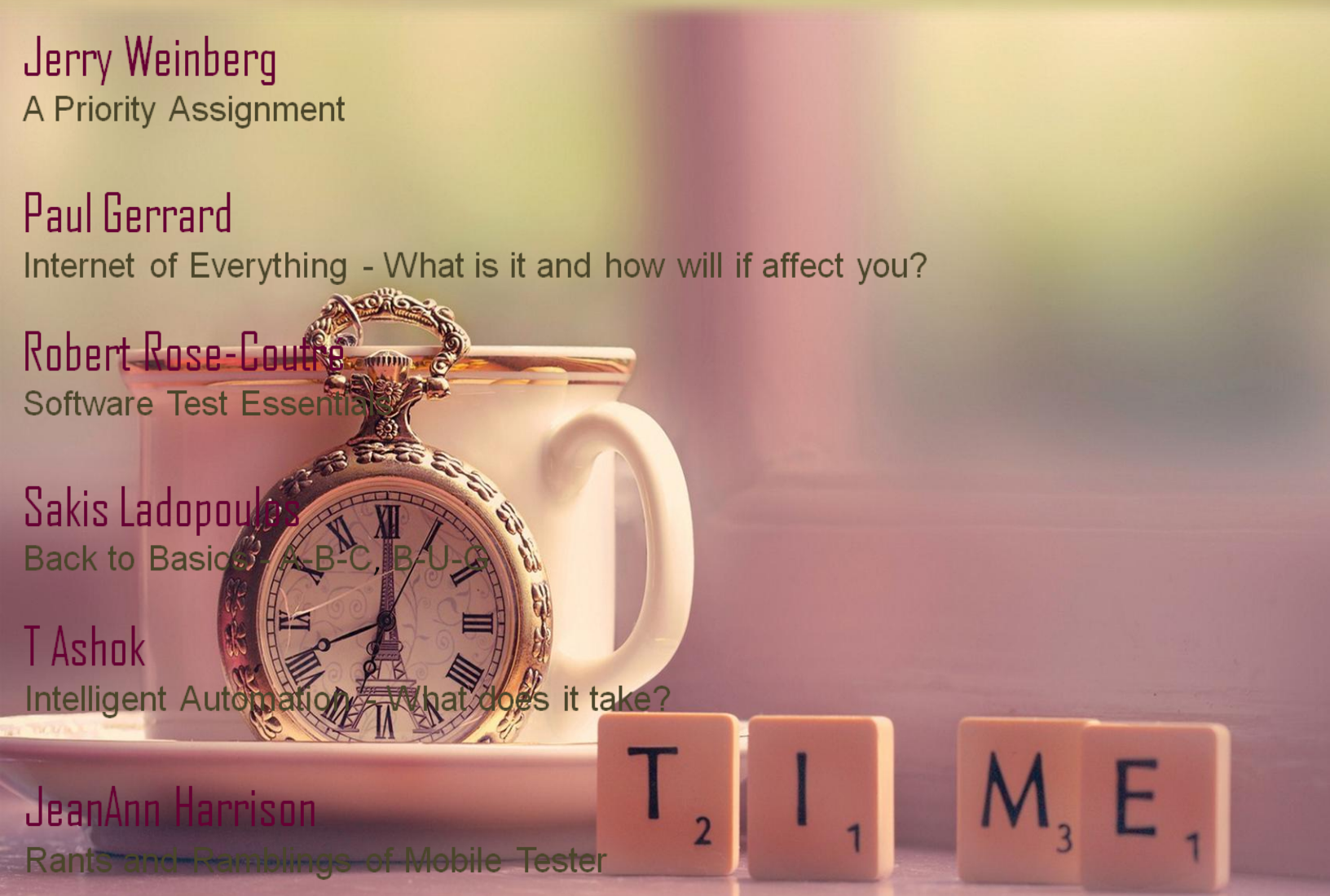
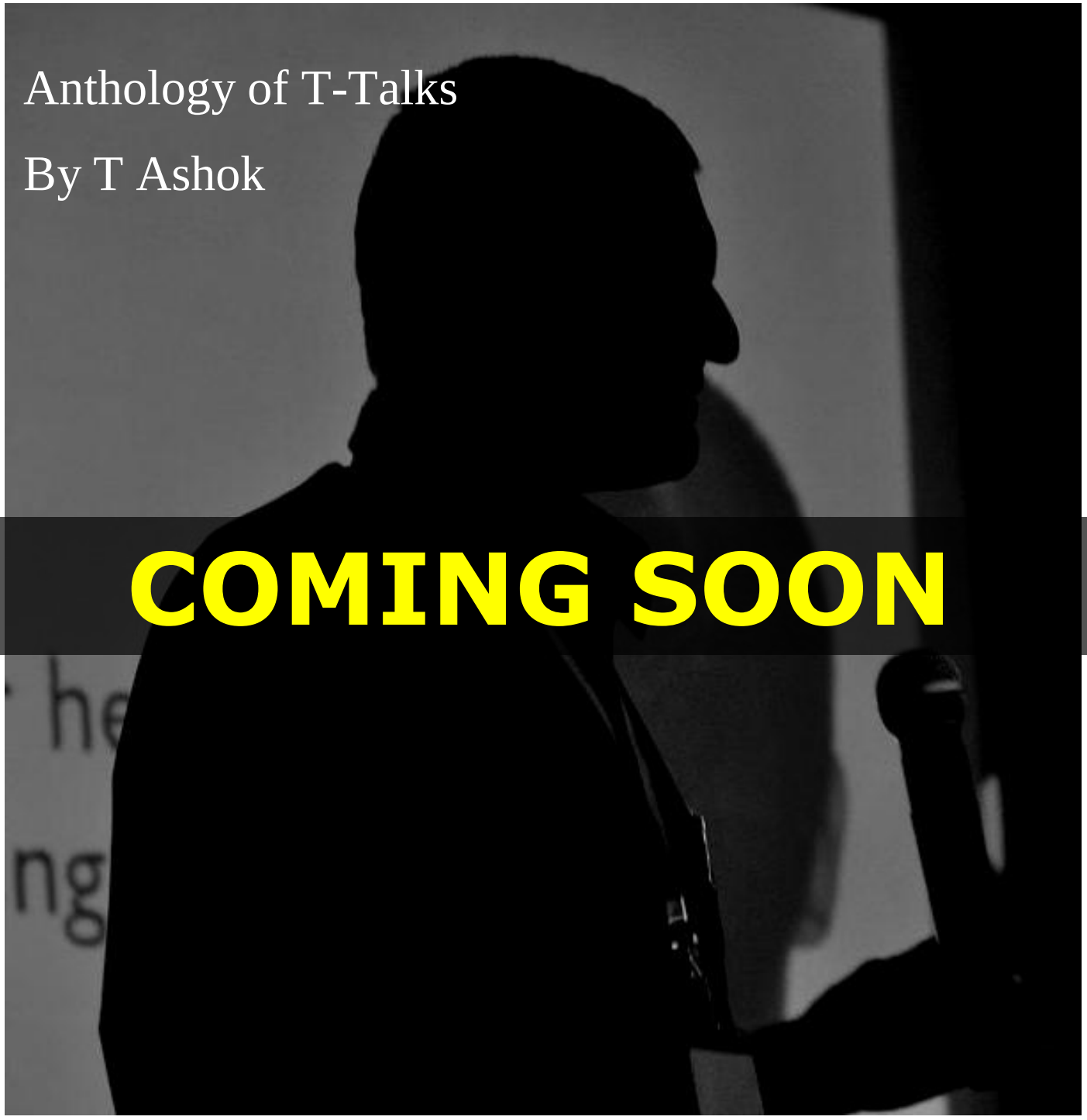
THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

MARCH+APRIL '14 | YEAR 4 ISSUE III



Musings over Tea-Time



Celebrating THREE years of Tea-time with Testers. Anniversary Special!

ANNOUNCING...

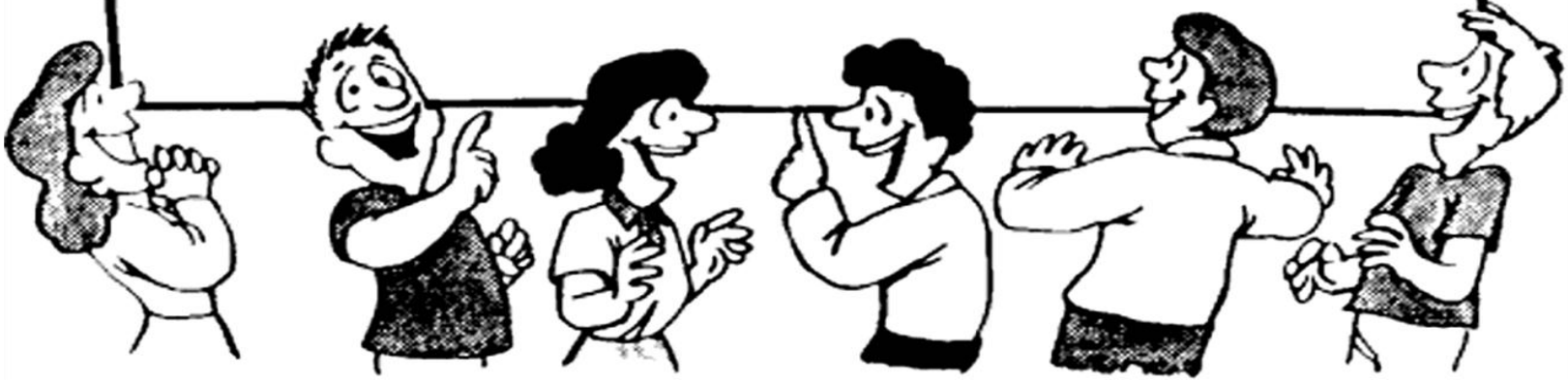
STATE of TESTING

LIVE webinar by Jerry Weinberg and Fiona Charles

Few months back we had conducted the **'State of Software Testing'** survey for year 2013. Results of this survey have been already published. But what do those results tell us about state of software testing?

Join this **LIVE** webinar to know what experts like **Jerry Weinberg** and **Fiona Charles** feel about it.

Click Here to Register



~ brought to you by ~

Tea-time with Testers in association with **PractiTest**

Asia Pacific's Largest Software Testing Conference

11th International Software Testing Conference

STeP-IN SUMMIT 2014 / Testing NOW

Interaction » Insights » Opportunities

JUNE 25-27, 2014 AT BANGALORE &
JUNE 18 & 20, 2014 AT PUNE & HYDERABAD

Call for Speakers

Submissions are invited for Speaking Opportunities & Best Publications

STeP-IN SUMMIT 2014 Best Speaker

Here's a chance to be recognized as a great speaker / contributor in the Software Testing space.

A great opportunity for:

- Industry Recognition
- Acknowledgement on the conference website, and through social media
- Complimentary passes to STeP-IN conferences for the next two years
- Cash awards
  - Audience Choice Best Speaker Award – INR 10,000/-
  - Jury's Choice Upcoming Star Speaker – INR 10,000/-

For more details visit: www.stepinsummit.stepinforum.org
email: papers2014@stepinforum.org

Produced by:
STeP-IN Forum
www.stepinforum.org

Hosted by:
QSIT
The Knowledge Corporation
www.qsitglobal.com

TEA-TIME WITH TESTERS

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

Created and Published by:

Tea-time with Testers.
Hiranandani, Powai,
Mumbai -400076
Maharashtra, India.

Editorial and Advertising Enquiries:

Email: editor@teatimewithtesters.com
Pratik: (+91) 9819013139
Lalit: (+91) 8275562299

Editorial

Look what April has in store for you!

Season is changing and spring has already started showing its colours, hasn't it? Then how can we not offer something more colourful and full of life to our readers this time again?

You guessed it right. Along with some awesome articles this month we have also brought two awesome, live webinars for you. Do not forget to join us in 'State of S/W Testing webinar by Jerry Weinberg and Fiona Charles' and 1st webinar from his 'Internet of Everything' series by Paul Gerrard. You'll get to know details of both the webinars as you read this edition.

Joel and T Ashok have continued to shower us with their wisdom in their regular columns. And you'll love what Robert and Sakis have contributed in this edition. My special thanks to them for considering writing for TTwT. And do I need to mention about JeanAnn Harrison again? I personally find her mobile testing tips and techniques very helpful and I am sure that many testers would have got benefited by her articles. By the way, she is hosting Mobile Online Summit soon and you may want to attend it.

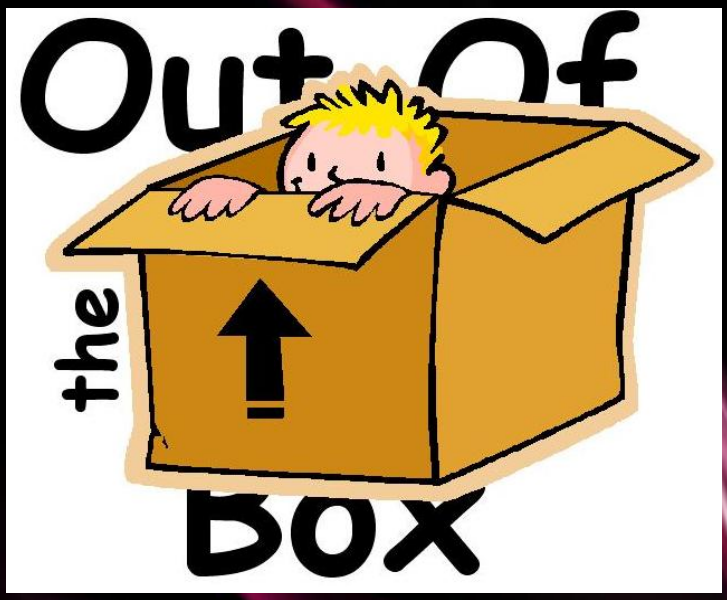
And before I wind up, let me tell you that there is one more awesome thing I am working upon. It will probably take some more time so please allow me an issue or two. This special task along with some other ones requires more time of mine hence we are clubbing our March and April issues.

That's all from my end this time. Enjoy your April until we meet again. After all, it brings a lot more with it and there is even more, much meaningful we can do than fooling people around ;-).

Yours Sincerely,

- **Lalitkumar Bhamare**
editor@teatimewithtesters.com

QuickLook



SmartBear SOFTWARE

Crossword by

Quality is delighting customers
QUALITY TESTING

Polish Championship in Software Testing will be held at the Polish National Stadium in Warsaw. The event will take place on June, 2-3.

This is the second attempt to find the best software testers in Poland. The first TestingCup took place in September 2013. More than one hundred testers faced Mr. Buggy - the application created especially for the occasion. The organizers have announced that this year's two-day championship will welcome doubled number of participants.

The championship consists in testing an application on the basis of specification delivered. Pure satisfaction of winning is not the only prize that winners can receive. In 2013 total award was 25 000 PLN (~6 000 EUR).

Everyone can participate no matter how experienced in testing he or she is. The competition is divided into two categories: individual and team. **Registration is now open.**

TestingCup is not only a competition for testers, but also a testing-orientated conference with both Polish and foreign speakers attending. If you want to become a speaker at the event, please contact TestingCup team before 10th of March at **testingcup@testingcup.pl.**

We encourage you to visit TestingCup website: **http://testingcup.com** and follow us on Facebook: **https://www.facebook.com/TestingCup**

A million dollar smile ?

Ask our sales team about our **Smiling Customer** ☺ programme

*Adverts starting from $100 USD | *Conditions Apply

Photo credits: Amir Eltahan

Tea & Testing with Jerry Weinberg

A priority assignment

One of the finest problem solvers we know got his start cracking codes for an agency whose very name was a secret. Over almost a decade of puzzle solving, he acquired considerable skill— a skill that was finally rewarded with a "PRIORITY" assignment.

His mission, with security name JACTITATION, was to "crack" the diplomatic code of "a small European power"—which happened to be an "ally" of his country. JACTITATION was to prove a two-year odyssey, but for over 18 months he seemed to be making no progress at all.

Finally, through meticulous tabulations, aided by the world's most powerful computing equipment, he began to be convinced that the diplomats were using a "book code"—a type that is virtually impossible to break.

Another six months of JACTITATION convinced him that the book on which the code was based must be a mystery novel. Two more months narrowed down the probable author. Then, at last, he found the book in the agency's comprehensive library of works of espionage and intrigue

—The Unpleasantness at the Bellona Club, by Dorothy L. Sayers.

He could hardly contain his eagerness to decode messages. Taking one he thought to be of extreme urgency, he began to translate the meaningless numbers into page, line, and word:

Page	Line	Word	Word at that location
112	25	7	Twenty
133	25	7	Three
157	27	5	Bottles
147	14	6	Scotch
19	5	7	Fifty
32	30	2	Nine
192	17	4	Wine

Figure 1. The secret message from a small nation.

"Twenty-three bottles Scotch, fifty-nine wine..." It was an expense account! Amused, he tried another message—another expense account! Two days later, he had translated 57 JACTITATION messages—every one an expense account! Two weeks later, our problem solver left the "intelligence" business for a career in teaching.

Before we close our opening, before we end our beginning, we should raise one more question that every would-be problem resolver should ask before seriously embarking on any problem:

DO I REALLY WANT A SOLUTION?

Though the question seems shocking, we've already seen a number of instances where the "solution" wasn't at all welcome, once it arrived. It may put the solvers out of a job, as in disarmament—though there we'd hope the other consequences were worth it. Or, as in the JACTITATION caper, the solution may be so trivial in its value that it makes us feel worthless.

We are trapped, quite often, because we've worked on a problem so long and so hard that we never really thought we'd solve it—so why worry about whether we want it or not?

Conversely, the problem comes upon us too fast for us to consider much of anything about the problem, let alone whether we want the solution. While a poor student window-shops without enough money in his jeans to buy a pack of matches, he dreams about owning a cabin cruiser or, at least, a pack of cigarettes. When he suddenly wins $100,000 in the lottery, his impulse will be to buy each thing he desires, though he may prove susceptible to seasickness or, at least, lung cancer.

Though many problems must be solved in haste, beware of someone pushing you to hurry.

Late in the resolution process, haste makes mistakes; in the first few minutes, haste makes disasters. Life is full of variations of the tale of the Fisherman's Wife:

The Fisherman finds a bottle entangled in his net. When the bottle is opened, a Genie escapes and tells the Fisherman that in return for freeing the Genie, the Fisherman and his Wife can have three wishes granted. The couple is, quite understandably, rather excited by the prospect. They sit up late that night discussing their dreams. In their exhilaration, they neglect their supper, so at about three in the morning the wife sighs and mutters, "I'm awfully hungry. I sure wish I had a sausage."

POOF! On the table is a delectable sausage, but the Fisherman is not pleased. "Look what you've done, you foolish woman! You couldn't keep your wits about you, so now we have only two wishes left. I wish that stupid sausage were hanging from the end of your nose."

POOF!

The reader, experienced in wish situations, can imagine how the third wish was used. At least the Fisherman and his Wife came out better than some of the other three-wishers, like the couple in the ghastly story, "The Monkey's Paw."

An old problem-solving saw goes:

WE NEVER HAVE ENOUGH TIME TO DO IT RIGHT,

BUT WE ALWAYS HAVE ENOUGH TIME TO DO IT OVER.

But because we don't always have the opportunity to do it over, we must do better. Put another way,

WE NEVER HAVE ENOUGH TIME TO CONSIDER

WHETHER WE WANT IT,

BUT WE ALWAYS HAVE ENOUGH TIME TO REGRET IT.

Yet even when we do want the solution itself, we may not notice there are inevitable auxiliary consequences that must accompany any solution.

Figure 2. I wish

One of the ancient quests of the alchemists was the "universal solvent," a liquid whose powers of disintegration could be resisted by no substance on earth. Like the quest for transmutation of lead into gold, this one seems to have been in vain. Too bad, though, because it would have been fascinating to know what they would have kept the solvent in, once they had it!

If we seek a universal solvent, we can hardly deem it a "side effect" that it dissolves any container we try to keep it in—presumably etching a hole through the center of the earth.

Yet we tend to regard "side effects" as the result of particular solutions. "They might not arise at all, and if they do, we can always refine the solution to eliminate them." How often does this naive attitude lead us into disaster?

If we set out to eliminate one cause of death after another, why are we surprised at the "side effect" of an increasing population of old people nobody wants? If we set out to eliminate causes of infant mortality, why are we shocked and dismayed when the overall population begins to blossom?

Part of the answer is the human propensity iov habituation: the successive reduction of response to a repetitive stimulus. Habituation allows us to cancel out the constancies in our environment, thus simplifying our lives. When something new appears in our little universe, it is most stimulating. After it remains a short time, offering neither threat nor opportunity, it becomes part of the "environment," or background. Eventually, it is cancelled out entirely:

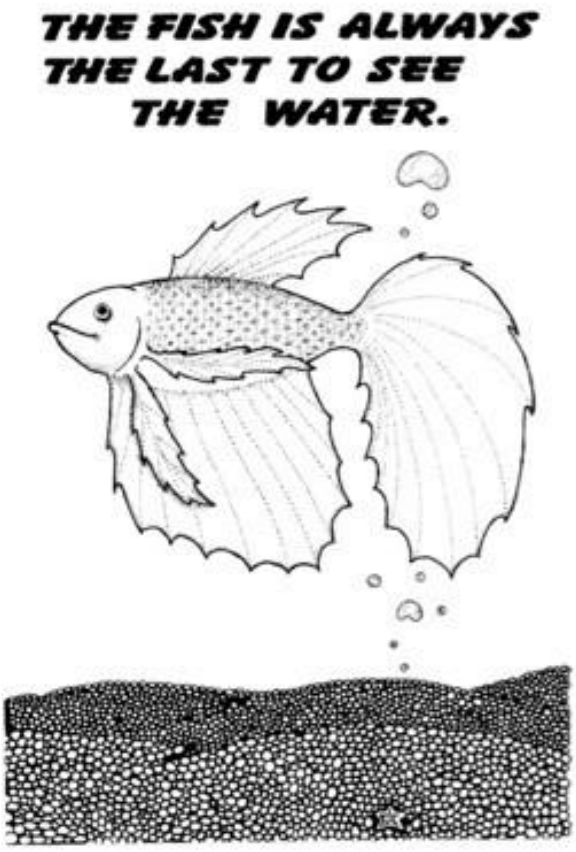


Figure 3. The Fish….

When we contemplate problems, items to which we are habituated tend to be omitted from consideration. Only when the "solution" causes the removal of the habituated element do we become startled. A most touching representation of this removal phenomenon was shown in Satajit Ray's movie trilogy, The World of Apu, when Apu's wife dies.

When he receives the news, Apu casts himself upon the bed, unable to move for days.

Director Ray shows him lying immobile for what seems to the viewer to be hours when, suddenly, his alarm clock stops ticking. Apu is startled out of his lethargy and the viewer—who has also been habituated to the ticking—shares the thunderous impact of the sudden absence. Only later do we realize that we have been made to share the shock felt by Apu when he realized—after her heart no longer beat —how much his wife had been a part of his life.

Like the filmmaker, the problem resolver is an artist dealing with imaginary worlds. Very early on—really from the very beginning—the problem resolver must strive to see the "water" in which the other participants unconsciously swim—the water which will be transmuted to sand when the "problem" is "solved".

Back To Index

Postscript

By becoming immersed in the problem, you, the resolver, risk yet another oversight.

Fascinated with the problem-solving aspects, you may neglect to consider whether you would morally approve of a solution. One person's sin is another's virtue. We wouldn't dare tell any reader that killing people is wrong, any more than we would dare tell a cannibal that eating people is wrong. Perhaps, even at some risk of appearing maudlin, we should quote Polonius in his advice to Hamlet:

"This above all, to thine own self be true."

To be true to yourself, in this problem-resolving business, you must consider moral questions before you get close to a solution, or even a definition, and thereby begin to lose your sensibility. Such consideration will never waste your time, for problem-resolving can never be a morally neutral activity—no matter how much it fascinates its practitioners.

Back To Index

Biography

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

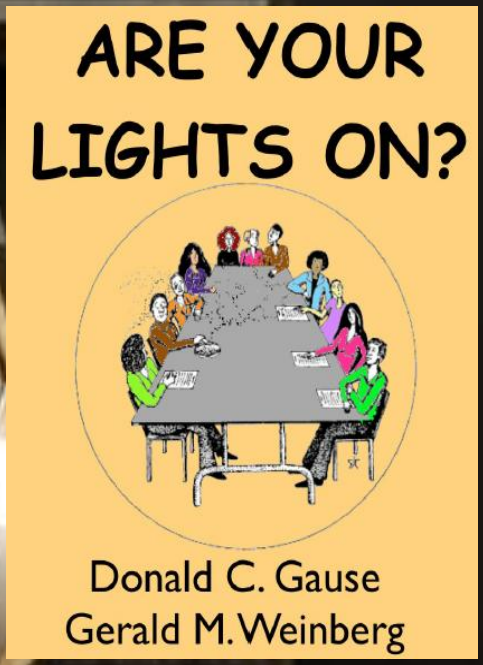
For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including The Psychology of Computer Programming. His books cover all phases of the software life-cycle. They include Exploring Requirements, Rethinking Systems Analysis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

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

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

Gerald can be reached at hardpretzel@earthlink.net or on twitter @JerryWeinberg

ARE YOUR LIGHTS ON? is one of the famous books Jerry has written together with Donald C. Gause.



ARE YOUR LIGHTS ON? has received great feedback from readers and we strongly recommend you to read it if you want to get 'problem solving' right, of course along with many other interesting insights that this book offers.

TTWT Rating: ★★★★★

Its sample can be read online here.

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

TEA-TIME WITH SMARTBEAR

About this column…

SmartBear Software not only provides testing tools to help development and testing teams accomplish their software quality goals, it is also a hub of information and news for the software testing industry. From workflow methodologies to discussions on industry practices and tech conference coverage, SmartBear has become a source for testers seeking quick access to a wide variety of content.

SmartBear's goal in creating this column in **Tea-Time with Testers** is to empower software testers around the globe by helping them become more informed about the current state of the software testing industry.

Developers Recognize the Need for Testing, But Still Devalue Testers

by **Gregory Mooney**

I had the pleasure of attending Mobile+WebDevCon last month in San Francisco. Being a fairly new and relatively small conference, I was a bit skeptical about how much I would be able to get out of it. Fortunately, the conference ended up having a good number of attendees, and the sessions were – for the most part – quite thoughtful and engaging. What really caught me by surprise was all the chatter about testing mobile applications.

That's right testers: this was a small community of *developers*, creating mobile apps, talking about mobile quality and the importance of testing. If I were to tag a theme onto this conference, it would absolutely be mobile quality.

If you're a commendable software tester, you understand that testing is not an easy task nor is it for the weak minded. To be an effective software tester, you need to approach the software philosophically, overcoming biases and understanding the context of each and every testing situation. Of course there is much more to it, but I am not going to go into it here. The point is that software testing is a skill that is not as easily attainable as some might initially believe.

So what am I getting at? I was a bit put off by some of the discussions about software testing. Of course, there were a few people who understood the importance of testers, but the general vibe was that testing is something anyone can do.

For instance, one speaker said you can just hire testers off of Craigslist, essentially devaluing the software testing occupation to that of a one-time handy man. I'm not sure what the intentions of that comment were, but I was definitely taken aback by this inference.

Since this was a mobile conference, I can only speak of the mobile developers at Mobile+WebDevCon, but there seems to be a serious divide around the importance of testers. I fully recognize this divide is hardly a new phenomenon, but the issue lies in that developers themselves are not putting their mobile apps under the same scrutiny as apps for other platforms, i.e. Web and desktop.

Roughly 80% of developers that I talked to at this conference said that they do both the development and testing themselves. That's like writing this blog post without an editor—the quality of my writing would be significantly degraded. I *could* try to edit my own work, but I know that I would likely overlook a typo or fumble over a phrase due to my own biases and partiality.

I understand that the lack of testing may be a resource issue, especially for smaller development houses, but is that an excuse when the credibility of your organization is on the line? You just need a few 1-star reviews on the app store to keep others from ever giving your app a shot.

The irony comes to life when the very same developers who tell me they don't need testers to do their testing turn around and insist that mobile app quality is the most important factor to the application's success – an assertion that is backed up by SmartBear's own original survey data.

Am I the only one who sees how crazy this is?

If the mobile industry is going to continue to evolve, testing needs to be at the top of every organization's task list. The mentality of "code now, test later… if there's time" is no longer acceptable. If time is a pressing factor go ahead automate some of the testing, but keep it in the hands of someone who truly knows what they're doing.

We've all seen the embarrassing results of nonchalant testing. Are you really willing to risk the future of your mobile app?


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

Software Test Essentials

by Robert Rose-Coutré

My Test and Quality Assurance roles have taken me through many schools of thought, many styles, many crises, many "aha" moments, many successful projects, and a few failed ones. After almost 30 years in my career (not counting my first 5 years of random non-career-related jobs), I've found that there are dozens of quality-centric roles that are not necessarily called "tester" or "QA professional." My Quality involvement has ranged from project conceptualization, to gathering and writing requirements, to requirements analysis, interface design, programming, production, production QC, project management, proofreader, editor, Help author, software tester, defect management, documentation management, process management, content manager, SDLC QA manager, all the way to retrospective facilitator. I'm not bragging, I'm just old and have had time to do a lot of things.

I asked myself a question: What ideas and practices are most important to remember, and especially important for new recruits to hear from the beginning. Today I've thought of six learnings that, in my opinion, summarize essentials of software testing and quality assurance. Some are general approaches, some are specific practices:

1. Adaptability to Context
2. Adaptability Meets Planning
3. Requirements
4. Use Case vs. Test Case
5. Smoke and Regression Test
6. Start-to-Finish

Adaptability to Context

Experienced software-quality professionals mix and match testing schools, styles, systems, theories, practices, to fit each new given situation. The "non-school school of thought" that embraces the mix and match approach, according to the context of each new project, is often called the "Context-Driven Approach" to software testing. Every new context expands your range of adaptability.

Adaptability Meets Planning

Though it may seem counterintuitive on the surface, planning is not opposed to adaptability. Healthy adaptability is spontaneity with discipline. Planning gives you vision and a path forward.

Planning includes market research, user requirements, resource estimates, business requirements, schedules, technical requirements, time estimates, schedules within schedules, staffing and expertise adjustments (warm bodies, knowledge management, alignment, etc.), and exit strategies.

Good planning helps you make more intelligent decisions to change the path forward as needed, and do it successfully. The best defense against things not going as planned is to plan thoroughly. A thorough plan equips you for contingencies to make swift and positive adjustments. Disciplined spontaneity can save you when plans don't go as planned, which happens in every project. But inadequate planning fragments your vision; you can't discern tangents and wasted motion from productive work. Bad planning leads to waste every time. Rigid non-adaptability also leads to waste, every time.

Requirements

Precisely defining your project is central to planning. Success or failure may depend on the quality of your Requirements. In addition to the obvious point that everything required of the software is documented in the Requirements, here are two tips I've found extremely useful and often forgotten:

1. Use "must"; not "should" — ensure every phrase denotes an element that is "required"—if you can't say "must," remove the item from the Requirements. Keep "nice-to-haves" separate.

2. Every Requirements statement must be "testable" — every phrase must map to a piece of functionality that must be present, which maps to a testscript that you can mark with a definitive "pass" or "fail."

Use Case vs. Test Case

There is a growing trend using Use Cases as Test Cases. Writing Use Cases takes a lot less time, requires fewer resources and less expertise. Use Cases are user scenarios—sequences of tasks performed on the software by a typical user. A Use Case is useful for one purpose, in User Acceptance Test (UAT), to verify the software works correctly in typical workflows. Use Cases will include alternate flows, but they are still confined to fairly normal end-user activities.

Test Cases cover the software more thoroughly and in more detail than Use Cases. Test Cases include every function that the software is capable of (or is supposed to be capable of); handling every type of data input/output, every expected behavior, every design item, and every class of defect. There are a

lot of Requirements that are not covered in Use Cases. But all Requirements must be covered in Test Cases.

To satisfy a Test Case, there may be one, two, or more testscripts. Ideally, testscripts have step-by-step, click-by-click instructions that any person off the street could see and instantly perform with no training. (But because of reality constraints, testscripts often assume knowledge common to the designated testers.) When the testscripts pass, the Test Case passes. When the Test Cases pass, the Requirements pass.

Earlier I said every phrase in the Requirements must be testable. Likewise, every part of a Test Case must be traceable to line items in the Requirements. For example, if I'm testing to verify that closing "print preview" takes the user back to the "print dialog box," then the Requirements document must state that closing "print preview" must take the user back to the "print dialog box." Otherwise, it's not required. Why test non-required functionality?

If I am testing the boundary of max characters allowed in a field, the Requirements document must state the max characters allowed in that field. Trace the test back to a requirement, or remove it from test.

To trace between Test Cases and Requirements, you might see something like this:

- Requirements excerpt: "… (0052) the notes text field must allow copy/paste functionality, (**0053**) the notes text field maximum characters allowed must be 250. (0054) The date field must …"
- Test Case item: **0053**: verify the notes text field maximum characters allowed is 250.
- Testscript section: **0053**: #1. enter 250 characters, click save (no error); #2. enter 251 characters, click save (returns error warning that the max characters allowed is 250).

Use Cases do not have this traceability; the 100 percent phrase-to-phrase, 1:1, reciprocal mapping to and from requirements. Test cases do.

Test Cases are for searching and exposing every type of error in the software (should be written by a seasoned SQA professional). Use Cases are comfort-factor UAT to ensure that no embarrassing errors will happen in the common workflow areas (could be written by customer service, technical sales, QA team, or product manager).

Smoke and Regression Test

Smoke test is for a quickie reassurance that new fixes basically worked, and did not introduce new side-effect errors. Failing a smoke test tells us the code has a fire to put out. But what do you do if smoke test passes and regression tests of the modified code pass. The dreaded reality is that where there's no smoke, there still may be a fire. Good coverage means regression tests beyond the area that was modified. Touching code can have unexpected effects in supposedly untouched areas.

Too much pressure on the pipeline to move forward at any cost, will cost you. Each release is a new opportunity for new hidden defects. Depth of regression testing will always be a judgment call on how far you take it. But keep in mind that a glitch can cost ten or twenty times more to fix after release than when it's found early. It is a wise practice (dare I say "best practice") to invest time and resources in regression testing to dig methodically deeper into "untouched" areas surrounding a fix.

Start-to-Finish

Finally, real QA means QA participates in all phases: project conception through launch. QA participates in storyboards, user/business/technical requirements, design, throughout the stages of SDLC, as well as conventional manual/automated release testing and UAT. The QA "hat" is a complex hybrid of many fabrics and textures that make the ideal QA professional a very rare breed.

Robert Rose-Coutré is a consultant with Trellist Marketing and Technology, serving as Digital Project Manager creating multilanguage country sites globally for DuPont. Robert started as a professional proofreader in 1986 and started managing teams in 1987. He has built, managed, and edited commercially successful websites; led SD QA efforts; helped launch and manage the Test/QA website stickyminds.com in 2000, directed publishing in several industries; and once in a while writes for enjoyment.

Robert can be reached at http://www.bobzeen.com/blog/contact.

Testing in 2014 - Automation, Mobile & BDD

by Luke Govier

I work as a Recruitment Consultant for Burns Sheehan, a leading IT & Digital Media Recruitment Agency based in the heart of the city. I specialise in sourcing Testers from Junior through to Director Level, and recruit for companies ranging from start-ups to some of the biggest companies worldwide. The purpose of this blog is to highlight the areas that I think will be instrumental for testers to further develop during 2014, and hopefully to give you a good overview of what I believe the most sought after skills will be.

Automation: Selenium Webdriver

One of the most noticeable changes over the past few years, and particularly during 2013 is the emphasis placed by companies on Automation testing. The most obvious of these tools is Selenium, in particular Webdriver. With the ability to write tests in the most popular programming languages and the fact it is Cross Browser and runs on Windows, Linux and Mac platforms, it has become the automation tool of choice for the vast majority of our Digital clients. With Selenium 3 on the way (good article on this http://seleniumhq.wordpress.com/2013/08/28/the-road-to-selenium-3/) it seems Selenium will be fundamental in testing for the foreseeable and so is certainly a tool worth adding to your repertoire during 2014.

Mobile: Automation Frameworks

Whilst automation is a lot more established on Web Based Applications, 2013 certainly saw the rise in Mobile Automation Frameworks. As it stands, there isn't a framework that has cemented itself as the 'go to' option, however there are a few staking a claim. The most obvious option is Appium, being that it uses the Webdriver JSON wire protocol and works on both IOS and Android devices. I've certainly seen an increase in the amount of companies adopting this after running Proof of Concepts on a number of frameworks, and with the Webdriver compatibility it is in a strong position. For those that are not completely sold on Appium however, Calabash framework also has gained a lot of traction over the past year.

Again with the ability to execute scripts on both IOS and Android devices, its main selling point is the fact is supports Cucumber and by extension fits into the BDD model that a lot of companies are adopting. Whilst there are other tools out there, such as Robotium (Android) and Frank (IOS) which do

have a good following, I foresee Appium and Calabash really taking off throughout 2014, and either would be a great addition to your tech stack.

Software Development Processes: BDD

Aside from testing tools and frameworks, there has been a marked shift over the past couple of years in Software Development Processes. The most notable current one is BDD, with a lot of companies attempting, albeit some more successfully than others, to follow this. Combining the general techniques and principles of TDD it has taken the idea of a Ubiquitous language from Domain Driven Design and made it a central to its theme. With the focus on allowing communication and collaboration between technical and non technical participants in a software project, the business value of following this method has started to become more noticed by companies. A number of tools have been created to help implement BDD (JBehave/Cucumber/Specflow), so it is certainly worth gaining exposure to these tools where possible.

What are your thoughts on the outlook for testing in 2014? Is there an automation tool that you think should be mentioned that hasn't been? Do you think different Mobile Frameworks will stake a claim? Do you think BDD is as valuable to an organisation as many believe it to be?

Luke Govier works for Burns Sheehan and is partner with some of the most exciting and innovative companies in the Digital Media World to help them find and retain the best Testing and QA professionals. These range from major blue chip companies through to SMEs and start ups.

He is the founder of the Linkedin Group Test & QA London and regularly posts his own blogs, industry events and articles in this group.

Back To Index

A Context Driven Awakening...

by Faiza Yousuf

It was 11th January, a cold morning in Karachi, and we were all excited and warmed up as this will be the launch day for OuttaBox! We have been working round the clock to bring it to life past 1 year, and have been shaping up its foundation and active ingredients.

The event took place at The Dot Zero, a fabulous Start-up incubator and a professional gathering point right at the heart of the city. For IT professionals in Pakistan "Start-ups" are still considered as a risky business, and when it comes to the IT sector the ball comes down to a whole new game.

OuttaBox, as the name shows reflects the two sides of this business; (a) OuttaBox as a new comer in the Tech Consultation business are willing to take the risk and (b)taking the challenge of Context Driven Business Approach!

We divided the event into three portions:

a. An introduction to OuttaBox and its business

b. A Teaser for our prime Training area called "OuttaBox Tester" a Context Driven Testing Workshop.

   And

c. Hitting the trendy IT brains with "Software Test Automation" and gave the audience of what's and how's of "Selenium"

Arslan Ali (Training Consultant at OuttaBox) took the stage with an introduction to "CDT Workshop". He presented the audience with what the baseline of training is and how that would turn them into real testers with the right coverage and finding those important bugs; He presented the ideology of being and "OuttaBox Tester" with a mix of Human and Technology. He also gave the audience of having the right tester's identity and how to differentiate you from the normal lot of techies. The audiences were presented with the approaches such as "Heuristics Test Strategy Model" and how oracles and heuristics are used to Discover and Identify bugs.

The stage was then given to Sufian Jawaid (Training Consultant at OuttaBox), the expert techy in Automation; with the inline foundation of Context Driven, Sufian presented a complete paradigm of Test Automation ideology and raised questions to the audience about having the right automation strategy. He then turns the tables around and took the crash journey into what is called "Selenium";

Selenium has caused a lot of hype in the past years regarding browser based testing; it has created sort of a needed trend for Testers and in Pakistan when it comes to trends, well, you need to follow them!

Finally the stage was given to Faiza Yousuf (Chief Consultant at OuttaBox); Faiza by profession has been involved in establishing and running Start-ups, her last one "Frontal-Labs" created some good vibes around.

What Faiza told audience that day was about the target areas for OuttaBox and TestersTestified (@testtified), the crazy meet up setting for the Testers in Pakistan, as the Gollum in Lord of Rings shows his obsession for the "precious" ring, Faiza presented "We Loves Testing – We Loves it".

OuttaBox represents a complete Paradigm for its business consultation services, which includes;

- People
- Product
- Promotion
- Process

  And

- Technology

By representing these area OuttaBox is actually targeting any problems, under any circumstances with it range of solutions, namely, Trainings, Promotions, Testing, Development and Re-structuring.

The event was then concluded with an informal photo shoot of the audience with the OuttaBox Team.

For further details on the event Pics and what OuttaBox is up to these days you can visit our FB page (https://www.facebook.com/OuttaBoxPK), or follow us at Twitter (@OuttaBoxPK).

Back To Index

Faiza Yousuf is a graduate and post graduate from NED University's CSIT Department. She is a certified Software Quality Professional from PIQC institute of Quality and a member of American Society of Quality. She is also one of the founding members of Software Testers Engagement Program (STEP).

Currently working as a Project Manager in a software company "Tectutive (Pvt.) Ltd." and heading both engineering and business engagement departments, she is also running a Context Driven Consultation Company "OuttaBox". Her recent training adventure "Testers Testified" is about creating awareness regarding Software Testing and setting a Quality oriented trend in the local industry.

The plan is to translate training sessions into international Software Testing Conferences and escalate to another level.

Introducing...

Quality Remarks

- a column to discuss your test management problems with Keith Klain!

So, are you having testing times lately?

Want to bring change in your testing culture and need help?

Have exciting ideas around testing but not sure about implementation?

Or looking for guidance and advice on your testing problems?

Worry not. Keith Klain is here!

In this exclusive column, Tea-time with Testers is offering you an opportunity to interact and get help directly from experts in industry.

Many thanks to Keith Klain for agreeing to offer his guidance through this forum.

How will it work?

It's easy. Just send us your questions on editor@teatimewithtesters.com and we will publish Keith's answers in subsequent issues of Tea-time with Testers.

Make sure to mention **Question for Quality Remarks** in your subject line.

Note: Tea-time with Testers reserves complete right to deny any question without giving any justification.

Keith Klain is the Chief Operating Officer for Doran Jones, a technology consulting firm specializing in software testing and agile development. With 20 years of multinational experience in enterprise-wide testing programs, Keith has built and managed global test teams for financial services and IT consulting firms in the US, UK, and Asia Pacific. Keith is a current member of the board of directors for the Association for Software Testing and was the recipient of the 2013 Software Test Professionals Luminary award.

Visit his blog at www.qualityremarks.com

Follow him on twitter: @KeithKlain


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

Internet of Everything – What is it & how will it affect you?

- part 1

by Paul Gerrard

There is an increasing amount of publicity, information and hype around the subject of the Internet of Things (IoT) and the Internet of Everything (IoE). What on earth are people talking about? Should I be interested? Will it affect me? What does it mean to me as a human being? What does it mean to me as a *tester*?

In this article series I want to explore what the IoT and IoE are and what we need to start thinking about. I'll approach this from the point of view of a society that embraces the technology. Then I will look more closely at the risks we face and finally how we as the IT community in general and the testing community in particular should respond.

In the first article of the series I will look at what IoE is and how it affects us all. It's important you get a perspective for what the IoE so you get a sense of the scale, the variety, the ubiquity, complexity and challenge of the technological wave that many people believe will dominate our industry for the next ten to twenty years.

Let me start the whole series off with what sounds a bit like science fiction, but will soon be science fact. John Smith and his family are an invention.

The Human Perspective

John Smith's family live in a quiet suburb. He is married to Alice, has two children, works as an engineer for the local authority, and is a keen road bike rider. The family recently moved into their new house and are still coming to terms with the technologies that have been designed into it. It seems that every day, some new feature of the ubiquitous, embedded technologies emerges. All of these features are described in great detail in the user manual – but who has the time to read the manuals for a house?

At 6.00am, the alarm rings on John's side of the bed. The vibration and sound was chosen because John is sensitive to it, but his wife isn't. John gets up, grabs his phone and moves quietly to go cycling. The house has already turned the central heating on in the rooms that need heating and as he moves around the house, lights come on automatically. The security alarm turns itself off. As he leaves rooms, a sensor detects this and lights are dimmed and turned off after a pre-set delay. In the bathroom, he uses the toilet, weighs himself, brushes his teeth and washes. Like all the other electronic devices in the house, the toilet, weighing scales and toothbrush are all connected to a central hub installed in the house. Who knows what data they collect?

He gets a drink and an energy bar from the fridge. The fridge beeps - the energy bars are running out – and it logs the removed items and updates the shopping list. John touches the screen on the fridge to confirm he is eating more than 6 per week to adjust the re-order level. The door to the garage unlocks as he approaches it. He puts the drink and energy bar in his rucksack, grabs his bike and pushes the bike out of the garage. The garage locks behind him, the lights fade, he pushes off and he's on the road.

His phone connects with the power sensor and odometer on the bike and the heart monitor he wears on the elastic belt around his chest. The phone logs the energy he uses, the distance travelled and his heart rate throughout the journey. In the meantime, his GPS position is tracked and saved to the phone. John is on a short route today as he needs to get to work early. His phone vibrates and alerts him – the weather later on is going to be wet – but he knows he'll be all done before the rain comes.

It is early, still dark, and the street lights are off, but automatically switch on just before he reaches them. The roads are still quiet and so, the street lights turn off again after he leaves them behind. At halfway around his route, he stops, eats the energy bar and take a drink. He logs the food and drink on his phone. John's route is circular and after a few more kilometres, he climbs the long, steep hill on the circuit. His bike reminds him not to exceed 165 heartbeats per minute and he settles into a steady rhythm. Today is meant to be an aerobic session. He speeds down the other side of the hill.

John returns home and as the garage door opens automatically, his phone uploads the statistics of the ride to a website that tracks all of John's exercise. As he puts the bike away, the phone reminds him that the rear tyre of the bike is 1000 kilometres old and might need to be replaced soon.

John goes to the bathroom to have a shower. His wife, woken by her own alarm – the radio – is just getting up. The kids are still asleep but they'll be waking up soon. A message from the school to the home hub warns Alice that they won't need their swimming kit today – the pool is closed for maintenance for 48 hours.

John dresses quickly and is ready to go to work. He says hello and hugs the kids as they sit and have breakfast. Alice will drive the kids to school a bit later. John kisses Alice, says "see you later" and walks out of the door to his car. The car unlocks as he approaches and he gets in after the seat automatically adjusts (as Alice drove it last night). He drives off. His journey lasts twenty minutes but on a bad day, it can take half an hour. The car, of course, knows the destination.

The car is in communication with the local traffic management network and the 'crowdsourced' cars on the network identify their location every few seconds. The network knows where the bottlenecks are and of course, so do the cars. John's car gave an audible warning to take the ring road, rather than the direct route today.

His car is aware, through networking and radar, of the location of other cars on the road and the path of the road itself. The car gives warnings of hazards on the road, even around blind corners and issues 'do not pass' messages when it would be dangerous to overtake.

The car offers to read out his messages – IMs, texts and Twitter, as he drives. It knows already which messages are relevant to him at this time in the morning. One reminds him of his meeting with a client at 10am at the coffee shop in the atrium of his office building.

As John approaches the barrier to the car park of his office, the barrier automatically rises to let him through. His office operates a hot-parking space regime, so the car knows where to find the space nearest to the office entrance that will get him to his allocated hot-desk quickly. The car directs him to a space, and for a change, John parks the car himself rather than let the car do it automatically.

As he leaves the car, it locks and disables itself, although an authorised valet service that operates in the car park can open his car, with a code, to clean it – although they cannot start the car or its services.

John enters the secure building but doesn't need to swipe a card or show a pass. He grabs a coffee from a machine (which charges his account automatically) and walks to his desk. The surrounding lighting changes imperceptibly as he sits down and arranges his papers. The movement sensor in the office space also adjusts the air-conditioning and humidity in his area so he has a perfectly comfortable working environment. All these facilities are controlled by the local building management system which is constantly monitored by the landlord's central systems.

John has interacted with hundreds of sensors, computers and devices and it is only 8.30 am in the morning. Time to do another day's work.

The Technical Challenge

In recent years, there has been much progress in developing connectivity between Internet-connected 'smart' devices for intelligent monitoring, remote sensing and control using advanced analytics and real-time processing. To date this has mostly been based upon IP and Internet-based communications but the pace of development means that soon, existing communications and networking technologies will be inappropriate, and probably inadequate, to handle the traffic generated by an Internet of Everything.

We are all familiar with the office- or home-based computer that connects to the internet through a wired or wireless connection. In recent years, the mobile phone and tablet technologies have extended the use of the Internet to people on the move. It is natural to assume that machine to machine (M2M) and wireless sensor networked (WSN) implementations are simply extensions of the Internet. But there are significant challenges to be overcome to make this a reality.

Firstly, many of the 'things' to be connected will be low-powered devices which need to successfully function for months or even years without attention or a re-charge. Secondly, these devices transmit relatively low volumes of data, but do so over lossy and noisy networks. The existing Internet protocol is not an ideal solution.

Currently, the web relies primarily on IPv4 addresses the HTTP and TCP/IP protocols. The IoT will be delivered with a variety of evolving standard protocols with unfamiliar names and acronyms like 802.15.4e, 6LoWPAN, RPL and CoAP. These emerging standards aim to integrate wireless networks of low power devices to the broader Internet.

IoE Standards are Required

Needless to say, the IoE will significantly increase the number of devices connected to the internet by a factor of anything from ten to perhaps one hundred times. It is clear that the Internet needs the more efficient, emerging protocols to deliver the promise of IoE. These protocols are still evolving and are competing for attention and adoption somewhat, but the hope and expectation is that the standards will be refined and mature in the next couple of years.

The challenge for the developers of the new Internet will be, 'which architecture, which standards to adopt?' The low-power devices and WSNs are proliferating but these local systems cannot match the performance of standard Internet hardware at scale. The Internet architecture is familiar and provides a choice of protocols such as HTTP, SMTP and is scalable but does not work so well with low-power devices in lossy networks.

So, the race is on to develop standards that can interconnect low power devices using different proprietary protocols with a seamless integration to the Internet. For the time being, most IoT deployments will be hybrid arrangements using bridging, gateways and middleware between the Internet and specialised, proprietary networks of low-power.

IoE Standards are Emerging

The huge increase in device numbers is driving the adoption of the IPv6 standard, which effectively allows for an infinite number of connected devices. The 6LoWPAN protocol enables IPv6 packets to be carried on Low Power, Lossy Networks (LLNs). A protocol for interconnecting such networks is also under development.

At the physical layer, the IEEE 802.15.4e standard will define the mechanism that allows for more resilient use of lossy networks (through channel hopping). At the application layer, the emerging protocol (that parallels HTTP) is a REST-based Web transfer protocol called Constrained Application Protocol or CoAP. It is similar to HTTP and uses its own URIs to identify resources and has, for example, GET, PUT and POST verbs, but it also has features that accommodate the low-power and energy consumption constraints of IoT devices.

The CoAP protocol implements a different dialog between internet devices (clients) and, for example, sensors. They are called 'observations'. The client sends a message to a sensor registering an interest in the output of the sensor. The sensor then sends data to the client without the need for long-standing connections or expensive polling by the client to the device. The nature of these dialogs is different to more familiar web, email or terminal-based dialogues between clients and servers.

Several other standards initiatives are in progress. Message Queue Telemetry Transport (MQTT) is being promoted by IBM. The ZigBee alliance is focusing on smart-home and smart-device applications. The Dash7 alliance has a different 'tag-to-tag' perspective. BACnet focuses on HVAC (heating, ventilation, and air conditioning), lighting and access control applications.

Back To Index

Now, if you are a non-techy, I must apologise for the headlong dunk into the murky waters of emerging technical standards.

The message I want to share is this: the efforts of the standards bodies makes for a dynamic and confusing state of affairs. It is not clear which standards will win the day. Whatever people are building now, it's unlikely their solutions will match the standards of tomorrow.

To be continued...

In this article, I have tried to set the scene of the wonderful future world of the Internet of Everything. Right now, it is a very confusing state of affairs, but clearly, an awful lot of effort and money is being invested in defining the standards and building business opportunities from the promise of the new Internet world order.

In the next article, I'll take a look more closely into the emerging risks of the Internet of Everything, from both a social and technical viewpoint.

Paul Gerrard is a consultant, teacher, author, webmaster, developer, tester, conference speaker, rowing coach and a publisher. He has conducted consulting assignments in all aspects of software testing and quality assurance, specialising in test assurance. He has presented keynote talks and tutorials at testing conferences across Europe, the USA, Australia, South Africa and occasionally won awards for them.

Educated at the universities of Oxford and Imperial College London, in 2010, Paul won the Eurostar European Testing excellence Award and in 2013, won The European Software Testing Awards (TESTA) Lifetime Achievement Award.

In 2002, Paul wrote, with Neil Thompson, "Risk-Based E-Business Testing". In 2009, Paul wrote "The Tester's Pocketbook" and in 2012, with Susan Windsor, Paul co-authored "The Business Story Pocketbook".

He is Principal of Gerrard Consulting Limited and is the host of the UK Test Management Forum and the UK Business Analysis Forum.

Mail: paul@gerrardconsulting.com | Twitter: @paul_gerrard | Web: gerrardconsulting.com

Have questions for Paul on this topic? Ask your questions and Paul will answer you in **LIVE webinar**.

That's not it; if Paul likes your question you can also win a **FREE** copy of

The Tester's Pocketbook

Send your questions to **editor@teatimewithtesters.com**

Webinar Details:

Date- April 19th 2014 at 8 pm IST (Indian Standard Time)

Click **HERE** to register

A-B-C, B-U-G

Types of Bugs & Testing Strategy tailored on them

by Sakis Ladopoulos

We testers need to find in software applications under test as many bugs as possible. We need to find anything that is wrong; that simple.

… but what is wrong? What is ***defined*** as wrong and how?

Before jumping into answers which include words like *specifications* and *requirements*, *usability* and *performance*, *testing phases* and *strategies*, let's disconnect from software engineering for a moment and go back to the routes of science …

Things are as they are, they do not need to be observed, measured, categorized or characterized so as to exist, so as to be. It is only the need of the observer that brings these actions into picture. The need of the human observer to understand, to forecast and to handle.

In order to measure, categorize or characterize something that is (the being), there should be a commonly understood and accepted reference. One minute, one mile or one pound has no sense if these metrics have not been defined a priori against a reference, a commonly known and accepted reference, so as to obtain logical meaning and physical value and to be used to measure.

So in order to categorize or characterize, we need to measure. In order to measure we need a metric and in order to create a metric we need a reference.

Going back to software engineering, what is the reference so as to characterize something as a bug in a software application?

There are mainly two major points of **reference** for the correct behavior of an application

1. The specifications defined for the software application
2. The common sense and common logic of a standard and common user

These two major points of reference help define (measure and categorize) what a bug is. Consequently we have two major categories of bugs:

1. The **non conformity-bugs,** where the application does not function as described in the specifications
2. The **logical error-bugs**, where the application –regardless of its specification- behaves in an irrational or unexpected manner according to what would be supposed to be expected by a standard and common user

It is important to say that while there may be bugs that belong in both categories, there is no significant correlation between them as they emerge from different references. As such, they are different in nature, they follow different patterns and they need to be treated differently in a test strategy so as to be defined, identified, isolated and terminated. In other words, you need different armory to exterminate different kinds of bugs.

Interestingly enough, there is rarely a clear separation of these two categories in test strategies, not to mention different methodologies followed so as to identify as many as possible from both categories.

We tend to focus as testers on the first category of non conformity-bugs while the second is usually not even referenced in test strategies and test plans and this is done, in purpose or by accident, for various reasons:

Well defined versus Undefined

Specifications of a software application are (or at least should be) clear and well defined. As such, it is straight forward to determine whether a certain behavior is a bug or not. Disputes and disagreements are usually solved when consulting the bible of the application - the specifications. Even if a certain behavior is not crystal clear in the specifications, a common practice is to enhance the specifications so as to make it clear. Just like in a legal system where the case law is continuously enriching the set of existing laws.

When it comes to logical errors-bugs we sometimes realize with great despair that common sense is not that common but it is a mean value of generally different opinions. Excluding cases where the error is obvious (system crash), the more people they look in a reported bug, the more different verdicts we get about the bug-or-not-a-bug. The actual horror in everyone's eyes comes if and when someone makes the fanatic's question: *where is this depicted in specifications?* Like if specifications is the answer to everything, the software's nostrum.

Finite vs Infinite

Specifications come usually in a format of one or more documents. Depending on the application, this documentation may be quite big, huge or ridiculously monstrous. It is never just a few easily readable pages. Even so, testers find comfort in the defined territory of specifications in comparison to the wild

unknown of what is supposed to be "common sense". Common sense is not only subject to interpretation but practically boundless. Even the simplest application may have in theory innumerous logical tiny errors. If we put them into picture, try to plan for them and expect them to be revealed, eventually plan becomes more vague and a project more expensive. Not to mention the difficulty to forecast what should be expected out of a boundless and subjective factor like "common-sense". Again one more good reason to set aside this category of bugs.

Should we plan and test focusing on logical error-bugs?

So what happens with this category of errors? If we seldom have a strategy for them, if we have not advanced and specific methodologies to follow how don't we constantly fail due to them?

In practice, we do find many logical errors while testing against the specifications. Especially in manual testing (and this is one of the reasons why automation and test-machines will never substitute completely the manual tester) against the specifications we find many logical errors and for sure the most important and apparent ones. Remember, how many times did you report a bug which cannot be directly linked to a specific test case or specification element? All these are logical error-bugs.

The ones that remain tend to be considered as "not-that-important" through management of expectations and not management of defects. They even give the feeling of a brand new application!! The customer acknowledges the smell of brand new with satisfaction when she/ he find such bugs, as long as they do not create issues in business continuity or in the neural system of end users. It is like the new pillow that is not yet fully comfortable or the brand new car which needs a bit more attention in the first few thousand miles. Minor flaws in the signature of brand new, not to mention that some of them may be treated with the question of horror: "*where is this depicted in the specifications?*" and thus be subject for a CR a.k.a. extra money.

For these reasons amongst others, logical error-bugs do not enjoy a special treatment in the software testing process. This does not mean by any means that we shouldn't evaluate and estimate their possible existence to the best level possible. Even if it is a black hole which seldom causes fatal issues in projects, this does not mean that should be fully neglected and set aside. We should know what we do not know. Put a tester which is not familiar with the specifications and leave him/ her to free-test the application along with the full-time testers of the application who may know by heart the specifications by now. If the new tester starts finding more bugs than the rest of the team, then … Huston we have a common sense problem!!

Note: This article is based on Sakis article "That's just wrong", published in Professional Tester, issue 24

Sakis Ladopoulos MSc. (TMMi Prof. certified) is a Test Manager with 8 years of hands on experience in forming, leading and managing through changes, teams of test engineers in IT and Telecom. Apart from Software Testing, which was his first job in Siemens AG more than a decade ago, he has also worked as internal auditor for Quality Management Systems and member of several work groups and committees for ISO and CMMI certifications having gained that way an oversight of Quality Assurance & Control area within the Telecom and IT industry.

He is also an occasional writer and speaker in software testing related topics.

Back To Index

Rants & Ramblings of a Mobile Tester

- by JeanAnn Harrison

When to and When not to Automate Your Mobile Testing – part 2

[click here for part 1]

Concept 4: Speed of Test

One of the most often missed concepts when considering if automation is applicable, is how fast can you run the test manually as opposed to planning out, writing the automated script and then running the script itself to get results? Although, just because the test might be a test you can manually do in 5 minutes doesn't mean you should dismiss automation for a particular test case.

This is why Planning out your Tests is such an important concept. But I see testers not consider planning and just want to jump right in and write scripts. This leap though is like jumping into a body of water where you can't see the bottom and testers can stall a project in trying to implement automation for the sake of doing automation.

Testers need to count time as a valuable resource which can make or break a company if the project is mismanaged.

So how long would it take to manually and visually test the rotation of one device and compare it to another configuration? Does the application or mobile website rotate when changing the direction of the tablet and then compare the display to that of the mobile phone? Depending on the application under test, you might be using the object position recognition for the automated script. Will that location change between builds, between projects, between versions? It's critical you think about how fast a test can be designed, developed and then executed before deciding to "just apply automation". Sometimes manual tests are just faster and do not need much documentation.

Concept 5: Maintainability

When considering automation, are you going to be able to maintain your mobile automated scripts? One of the biggest problems with mobile as I have previously mentioned, little is known for requirements, especially with the early releases of the mobile application or website. As the application evolves and the company receives production level feedback, new requirements emerge and Version 1.2 could look very different to Version 2.0. Most of those automated scripts written from prior versions won't apply, especially those using position as a way of recognizing objects. For those who believe in automating all or most of your mobile app testing, what kinds of test coverage do you feel you get? Do you cover temperature and behavior of the application in various situations? What about charging the device and testing how the CPU speed is a factor of accurate data produced by the mobile app? Are you testing the network communication of the software and the internet with regards to other hardware and firmware conditions? A great test to consider automating would be testing out the time change twice a year which primarily occurs in Europe and North America? This test is important for any application utilizing the device's clock and since the software for this function would rarely change, maintaining such a script would be relatively simple. Yet, if the testers are not planning out their tests prior to testing, they might miss this automation opportunity.

Concept 6: Regression and Functional Testing

Regression and Functional testing are the obvious choices for Automation but with mobile, sometimes, the road is not so clear ahead. However automating as much of the functional behavior is a great way to start but with any mobile project, you will find drastic changes going from build to build, release to release and version to version. When developing your mobile regression tests, keep them simple and modular so you can better be prepared for the evolution of the mobile software as new requirements emerge.

Another point about automating the functional testing, consider the most straight road with regards to end to end while deviated paths can be automated on their own but added as part of the main test suite. This way you can easily update without losing the integrity of the main purpose of your application tests.

Concept 7: Testing Beyond the GUI

I often talk about "Testing Beyond the GUI" because far too often testers, especially those new to mobile feel testing the GUI is "good enough". But when you start mobile testing, you need to go by the requirements. Little is documented about the interdependencies between hardware, firmware and software because those who create and write the requirements are not aware of the interdependencies within the entire system of which the mobile application is a part. With mobile devices containing so little resources, especially for mobile native apps, the tester needs to do a lot of testing to create various hardware conditions and combine with how the software uses and utilizes drivers. Here is where Exploratory Testing becomes necessary. There is much to observe, to be aware of behavior and display while setting up your test conditions which include hardware and firmware. You can't automate all of these kinds of tests.

There may be some tests you want to automate once you have found some patterns and know what to expect. But, how do you know what to expect while charging a device from a dead battery, engage the GUI which speeds up the CPU and increases the temperature? Does the software provide an error when the device gets too hot? Does the software recover once the battery cools off? Is there any sort of recovery? What do your stakeholders expect with error recovery? Remember, a device's battery is in a much smaller, confined space and closer to hardware like the cell modem which can be damaged unless there is some sort of check in the software. Developers and testers should explore to find out how the CPU speed, charging the device will affect how hot the battery gets before damage is done. Data can become corrupt due to unstable conditions. So, until you KNOW the boundaries, you cannot write a script with expected results.

Concept 8: One Size Does Not Fit All

Why do testers think for every application, mobile or not, there is one existing commercial tool which will allow them to do all of their testing? After having been testing in the mobile space now for 7yrs, I have come to realize each project is different, each application is different and my approach must be a new learning experience with each iteration. With some of the mobile testing projects, I've had to use various tools available, some open source, somecommercial and even create my own tool (actually I received valued assistance from developer teammates). You might find a tool to use for functional testing but doesn't have the capability to give you memory usage or CPU speed or temperature readings or battery voltage numbers, while you use the software.

This may require a need to find appropriate tools for the test. To find the right tools, you need to understand what kinds of tests you want to perform. What results are you looking to provide to your stakeholders? One tool will not give you the test coverage results your stakeholders and users demand.

Finally, working closely with your developers to better understand the architecture of your mobile software is key to mobile testing success. Work as a team to flush out more information to provide to the stakeholders. All together you can achieve more accurate requirements, better design and more thorough test coverage. With the complexity mobile software, testing is and becoming more complex as well. If you follow through on these concepts in applying automation, the more successful of a mobile tester you'll become. And NOW you know the answer to the "what is the best tool for mobile testing".

Thank You goes out to James Bach, Karen Johnson, Julian Harty, Jon Hagar and Dr Philip Lew for influencing my own learning process. A special thank you to all software testers who engage conversation, ask questions, provide a different perspective for me to consider which helps me to push further.

To be continued in next issue...

Jean Ann has been in the Software Testing and Quality Assurance field for over 14 years including 6 years working within a Regulatory Environment and 7 years performing mobile software testing. Her niche is system integration testing with focus multi-tiered system environments involving client/server, web application, and standalone software applications. Mobile software testing includes mobile native apps, mobile hybrid apps, mobile web applications and mobile websites.

Jean Ann is a consistent speaker at many software testing conferences, a Weekend Testing Americas facilitator as well as making guest appearances. She is always looking to gain inspiration from fellow testers throughout the software testing community and continues to combine her practical experiences with interacting on software quality and testing forums, attending training classes and remaining active on social media sites.

Back To Index

Online Mobile Summit

(5/14/2014 & 5/15/2014)

Has your role evolved into mobile testing tasks? Would you like to learn some of the latest mobile testing techniques? STP has lined up some of the most professionally experienced speakers for this online summit.

Mobile Testing is one of the most complex and misunderstood challenge for testers. Project planners, business analysts, and other stakeholders of the project rarely know the limitations of how a mobile application behaves on various devices. Even proprietary devices where the hardware is built by the same company, there are still unknowns for how the software behaves. How does the customer use their device and software? Is the application meant to be used in a relatively stationary position? Or

We have created our course content to include:

- Strategies for the four types of mobile software projects
- Apply testing techniques for automating mobile software
- Applying an error taxonomy to discover patterns in the mobile and/or embedded software applications
- Discover & apply User Experience testing techniques

Attendee takeaways:

- Development of test strategies based on the mobile software project
- Application of Automation for Mobile testing projects
- Performance testing practices for mobile testing projects
- What user experience testing means for mobie software projects
- Exploring patterns in bugs of mobile & embedded software to form stronger test designs.

Join us for an Online Mobile Summit on 5/14/2014 & 5/15/2014.

Follow us for registration details on http://www.softwaretestpro.com

Speaker list (no particular order):

- Scott Barber, Performanct Consultant of PerfTest Plus & SmartBear Software
- Fred Berringer VP of Product Management at SOASTA
- Raj subramanian, IT Systems Test Engr at Progressive Insurance
- Parimala Shankaraiah, Head of Startup Test Lab & Academy at Moolya Software Testing Private Lmtd
- Phil Lew, CEO of XBoSoft
- Jon Hagar, Embedded & Mobile Consultant of Grand Software Testing

Happiness is....

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

Like our FACEBOOK page for more of such happiness

https://www.facebook.com/TtimewidTesters

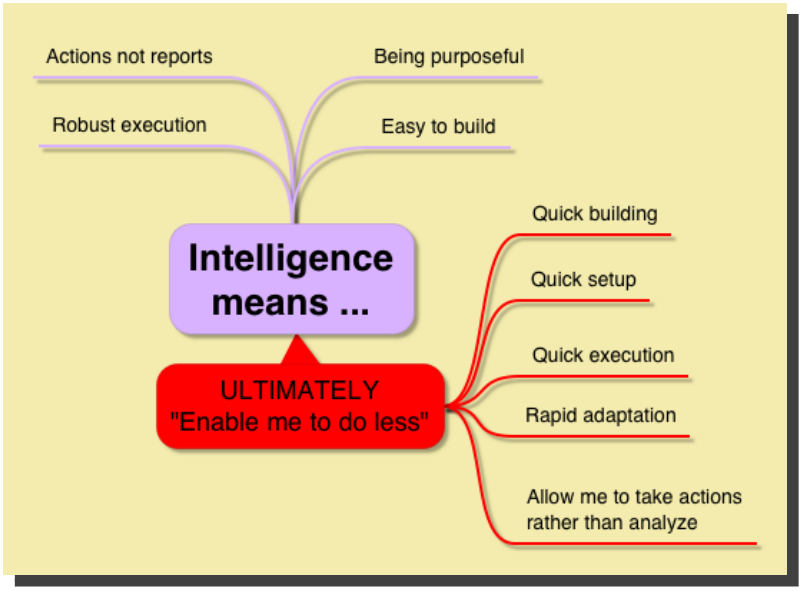

Call for Articles !
Have you got something to say?
yes, we are listening you...!!!
"Tea-time with Testers" firmly believes that one of the best ways to improve upon software testing is to listen to the lessons learned by others and their experiences too.
So, if you have an interesting story that you'd like to share with the world, contact us at teatimewithtesters@gmail.com.
Submit your articles, stories, thoughts around software testing.
now its your chance to be heard...!
Click HERE to read our Article Submission FAQs !
sciencephotogallery

Intelligent Automation - What does it take?

As we mature, automation needs to become smarter and intelligent to enable us to make superior decisions faster. It is no more about being a servile appendage that assists in doing things faster. It is about a leap from mindless repeated testing to continuous health assessment, to provide valuable information about the state of the system, to enable us to do less, yet deliver to meet the continuously increasing customer expectations.

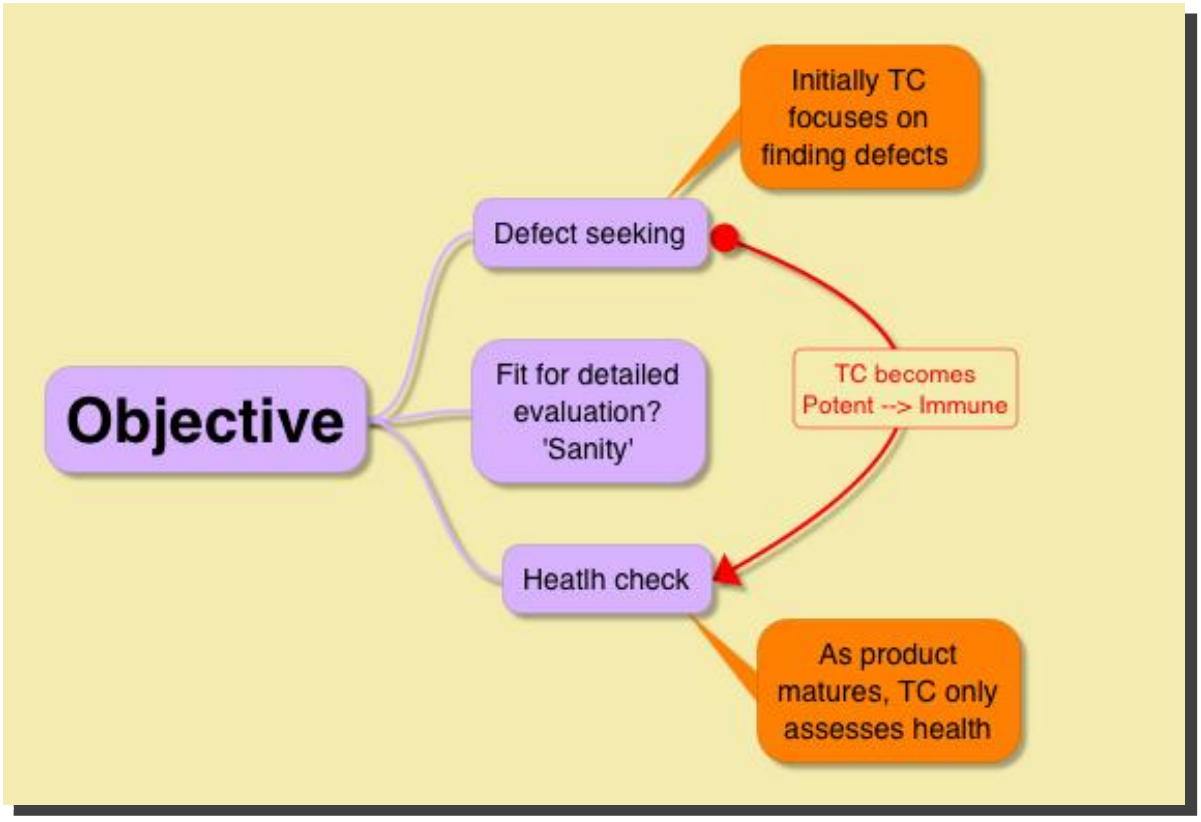
Intelligence means scripts that easy to build & maintain, scripts that are purposeful enabling me to clearly identify the problem, scripts that are resilient to ensure that maximal number of scripts executed in a run and finally scripts that are intelligent enough to analyze outcomes and suggest actions rather than report test outcomes as a report.

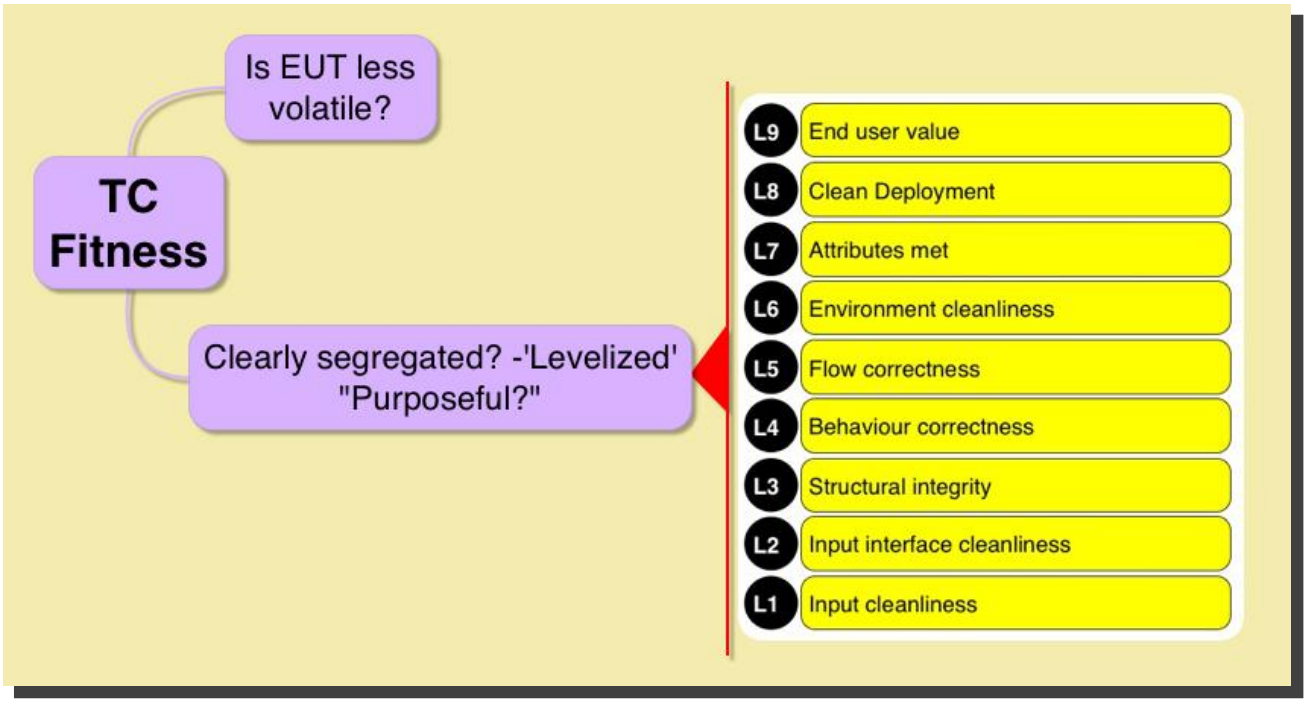
Ultimately intelligence is about "Enable me to do more with less" - Smarter. Cheaper and Better. Building scripts quickly, adapting them to newer versions quickly and cheaply, being smarter by analyzing outcomes rather than dump large data into a report. It is no more about fighting and solving technical problems, but to graduate to a higher level of delivering "Do more, but do NO more".



Let us step back for a minute and examine the objective of automated testing. Initially the test scenarios and the corresponding scripts are 'defect seeking' in nature focused on uncovering defects.

As the system matures with time, the 'potent' test cases becomes 'immune' and objective shifts to health check rather than find defects. Intelligence requires that we are able to diagnose the outcomes and display the health of system clearly to instill confidence rather than present data.



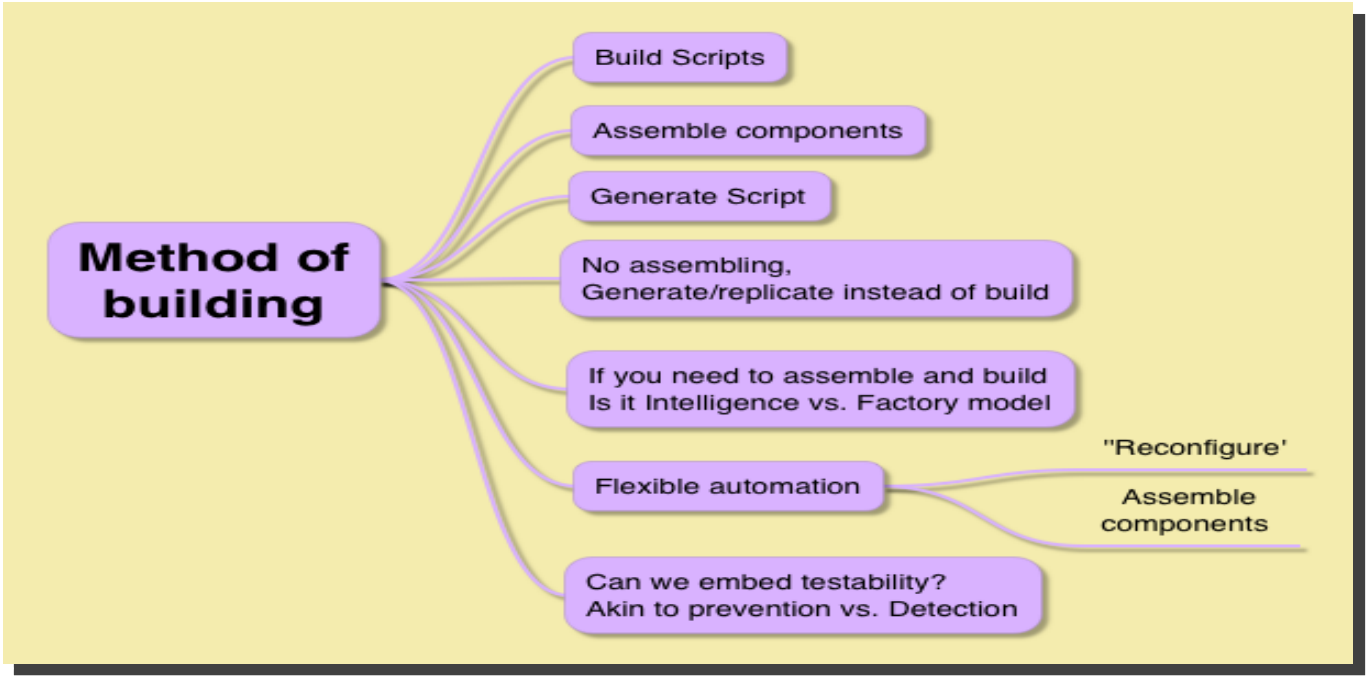


Now what is the role of test scenarios in intelligent automation? It would be desirable to ensure that scenarios be not very volatile.

And multi-faceted test scenarios that can uncover various defect types make the script inherently complex slowing up the ability to build rapidly and adapt quickly. It is essential that the scenarios be analyzed be for fitness and broken down into multiple levels (In Hypothesis Based Testing, there are NINE quality levels) so that the scripts are purposeful and small.

Now what would it take to create scripts intelligently? Other than handcrafting, we have various choices:

1. Build them by assembling components
2. Do not build at all, generate them
3. If it does not need to handcrafted, then ensure the framework is smart so that require less intelligence to build them i.e. factory model to building.



To enable smartness, the framework structure needs to be flexible. Only then can the script can be adapted quickly by "reconfiguring the connections" or by "re-assembling/re-wiring components". And to it all, can embed intelligence of testability inside the system, rather that have external intelligent scripts that assess the system. This would result in a system that can assess itself, the highest form of intelligence!

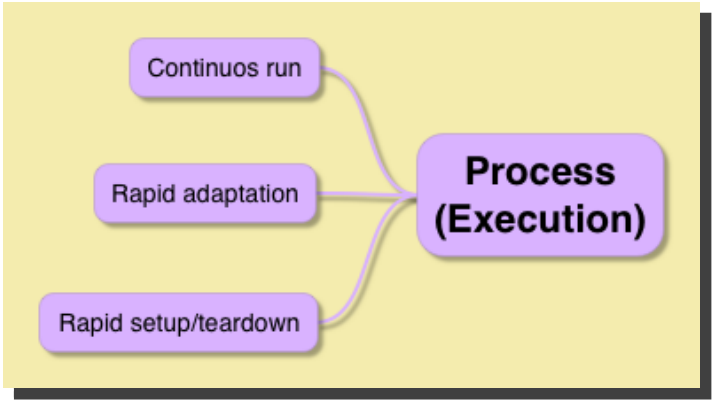
Moving from creation to execution, what does it take to do intelligent execution?
(1) Run as many scripts without stopping, intelligently 'jumping over obstacles' so as maximize the number of scripts executed. The obstacles posed may be defects, environment or setup issues
(2) Rapid setup/teardown to create the necessary environment and adapts as needed to minimize issues that prevent the script to run
(3) Finally rapid adjustment to adapt to new test environment.

Lastly let us appreciate the role of intelligence in test outcomes. Automated testing typically generates detailed reports that require deeper analysis to extract information. Intelligent reporting is about minimizing this, about presenting crisp descriptive analysis rather than voluminous outcomes.

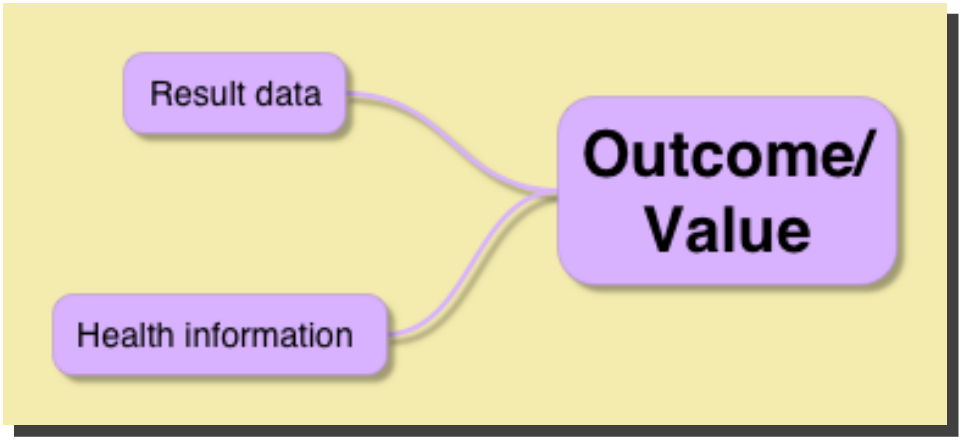
So what does it take to make intelligent automation?

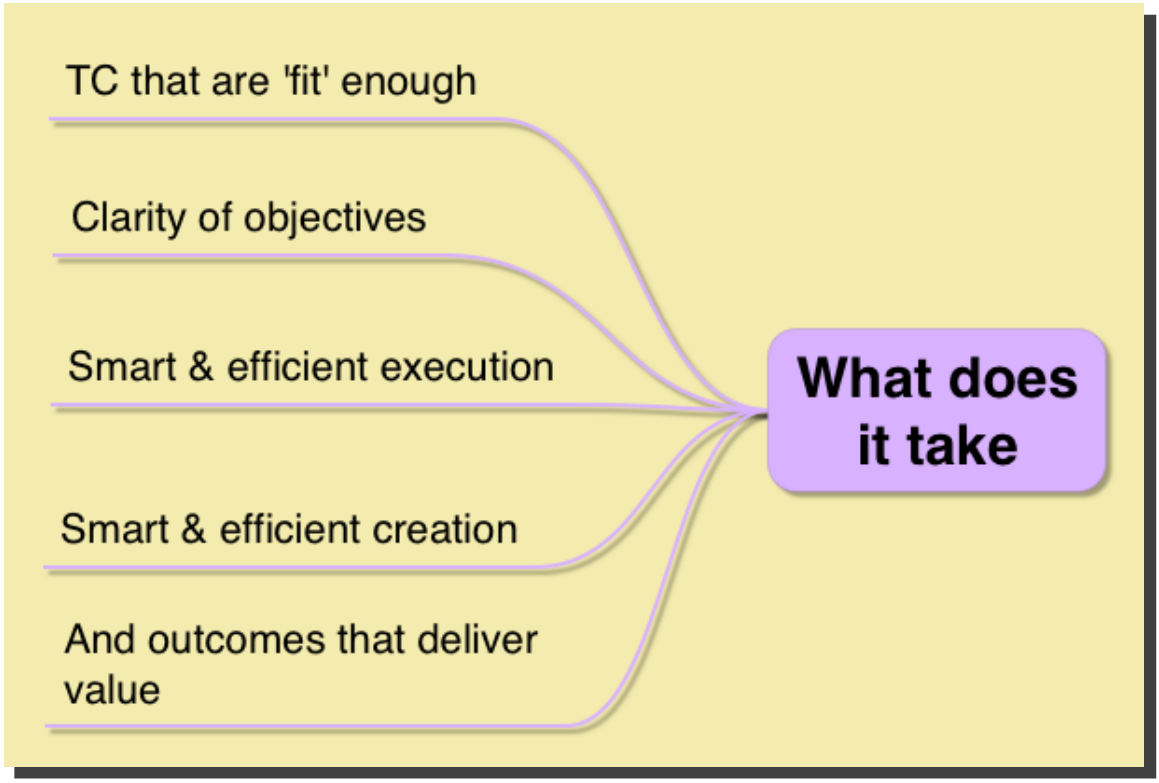
The mindmap alongside summarizes this.

Fitness of test cases, 'Levelized' test cases that are purposeful, good 'execution design' to maximize runs, rapid script creation, rapid adaption and crisply analyzed outcomes .



Intelligence is doing as less as possible to accomplish more. Ideally it would be wonderful to use the intellect to full capacity and not do anything physically anything at all! Like Sherlock Holmes who sitting still in his study, smoking his pipe could figure who the culprit was, while Inspector Lestrade was busy running around!





You are intelligent. Ensure that it rubs on your script too. May you work less!

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

testing intelligence

- its all about becoming an intelligent tester

an exclusive series by **Joel Montvelisky**

The lines are getting blurred

I hear about this all the time when I talk to PractiTest's users. You can read about it in the blogs and tweets of our fellow testers. People in QA conferences are talking about it more and more…
The trend is clear.

For most testers this is not an easy change.

At the beginning it frightened me. Then I learned to live with it. And, as I immersed myself into this new reality, I started to realize the incredible range of opportunities and improvements it offered both to development teams in general and to testers in particular.
What am I talking about…?

The fact that today ***the lines that separate between developers and testers are getting blurred***, resulting in more and more testing tasks being done by the developers within our teams. This is not happening in your team… Or is it?

The first reaction I get when I bring this subject up with fellow testers is that in their company this is not happening.

Then, when I ask if programmers are doing more tests today than they did 3 or 5 years ago the faces and tones suddenly change and become less secure.

Everywhere you look at developers are getting more involved in testing. They are writing more unit tests, Continuous Integration practices are common in many organizations, there are more programmers walking up to testers with questions about testing scenarios, etc.

The "virtual wall" previously used by developers to toss the product over to the testing team now has many doors and windows that both teams use to communicate and even collaborate.

What does this mean to the organization?

For starters it means that programmers need to learn how to test.

Testers get a chance to teach our development peers that testing (or at least good testing) is not a trivial task of randomly pressing buttons on the screen or typing "asdf" into text fields to ensure you can enter data into the system.

We've become testing mentors, explaining about testing scripts and testing data, reviewing positive and negative scenarios, explaining about pairwise testing and the testing heuristics we use.

Additionally, this change also means that our project teams now allocate time for developers to test their code. Even if this sounds trivial at first it is a big mental shift in the way companies prioritize their tasks, especially when projects get close to their deadline and time becomes the most scarce resource in the team.

What does this mean to us as testers?

I have already written that we've become mentors to our development peers. But does that mean that we are actually teaching our work to others in order to be relieved of our duties in the near future?

At first this is what frightened me, not personally for my future, but for the future of our career as a whole.

I was afraid that testers would become another VHS player or another Fax Machine, useful in the past, but doomed to extinction as times and technology advanced.

But then I realize that this was only partly true.

It is true that part of our job is being migrated to others in our organization, but it is also true that we are being given more responsibilities within the team and also being offered the chance to influence more on our process and the outcome of our projects.

In many places testers today have become strategists and advisors, working from the early stages of the project in order to assess and plan the overall development approach of each feature based on its requirements, risks and also based on what will need to be tested and how.

We've also been given more responsibilities to work with users both before as well as after the product has been delivered to them.

We are also being asked to perform more technical tasks, planning and architecting large parts of the automatic testing framework that will serve both developers and tester in our work.

Depending on the type of environments and companies we work in, testers are being given additional tasks. For example, in many SaaS companies (such as PractiTest) testers are tasked with many of the deployment and production monitoring operations, as well as troubleshooting issues happening to customers in their real-work environments. As a friend of mine put it some weeks ago, it is not about whether a bug is happening or not, but about how many times it is happening and to what percentage of our users…

And obviously, we are being asked to compile all the information that relates to the project and its quality status. Giving a live, clear and concise overview to the rest of the team on the state of the product, the status of the process and the ways in which both of them can be improved.

What is the next step…?

I am not sure what the next step will be, but as I said before the trend is clear and the lines between testers and developers are getting blurred.

For the good or the less good (let's keep optimistic) this is happening and it is something we all should be aware off in order to be prepared for the changes that will continue coming in the future.

Are you seeing this in your company? How is this changing the way you work today?

Share with us your experience to understand how this new reality is changing your life and your work as a tester!

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

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

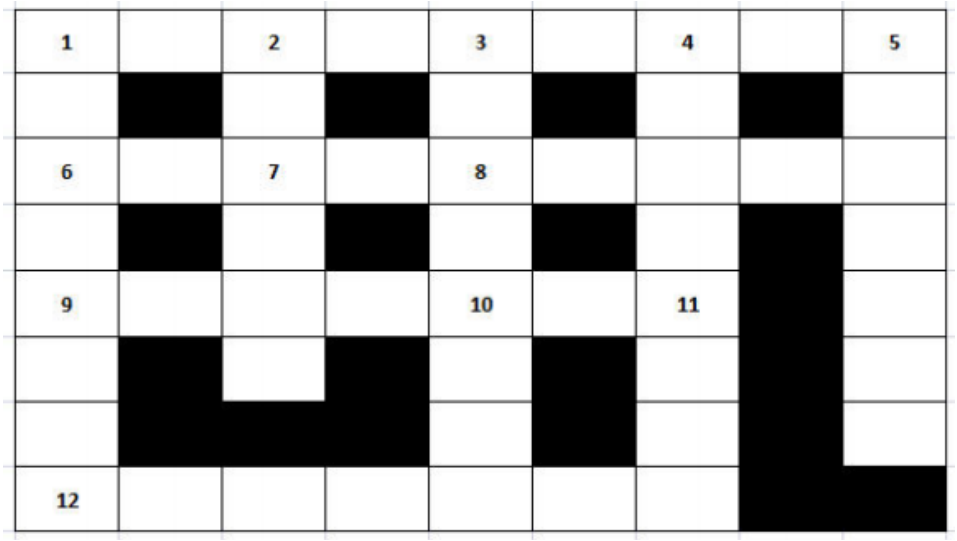
Today Joel is the Solution and Methodology Architect at PractiTest, a new Lightweight Enterprise Test Management Platform.

He also imparts short training and consulting sessions, and is one of the chief editors of ThinkTesting - a Hebrew Testing Magazine.

Joel publishes a blog under - http://qablog.practitest.com and regularly tweets as joelmonte

Claim your Smart Tester of The Month Award. Send us your answer for Crossword b4 25th April 2014 & grab your Title.

Send -> editor@teatimewithtesters.com with Subject: Testing Crossword



Horizontal:

1. Which company partners with Wipro to Accelerate Enterprise Business Innovation (9)

6. It is a suite of tools specifically for automating web browsers (8)

9. It is an open source and free load testing software, targeted mainly at web services (6)

12. It is a cost-effective, completely self-service Web site load testing tool, first name (7)

Vertical:

1. It is a web-based test management system that facilitates software quality assurance(8)

2. The set of all possible inputs, in short form (2)

3. The first executable statement within a component, in short form (2)

4. It is a distributed load testing tool (5)

5. A sequence of executable statements within a component (7)

7. It is the process of putting demand on a system or device and measuring its response, the first word (4)

8. Testing a system or application using negative data, in short form (2)

10. It is the last phase of the software testing process, in short form (3)

11. It is a GUI testing tool for Java swing-based applications (3)

Click here for answers to last month's crossword

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.

TEST ISTANBUL 2014
www.testistanbul.org

Mobile Testing: Testing on the Move

May 29-30, 2014
Sheraton Istanbul Maslak Hotel

Early Bird Fee until April 11, 2014: ~~€450~~ €400 + VAT

For registration: www.testistanbul.org | register@testistanbul.org | +90 212 290 76 62

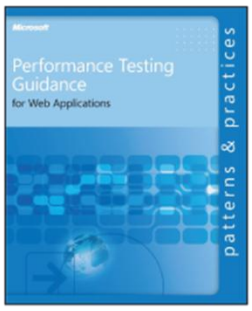
Keynote Speakers:

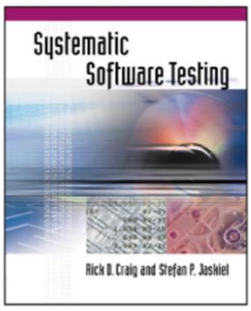
Scott Barber
Keynote Speaker
Author and
Chief Technologist

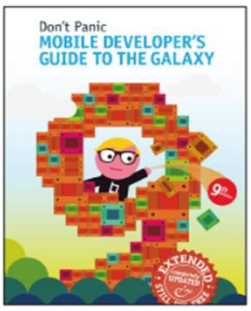
Rick Craig
Keynote Speaker
Consultant,
Lecturer and Author

Julian Harty
Keynote Speaker
Author, Editor and
Test Engineer

Dawn Haynes
Guest Speaker
Consultant and
Senior Trainer







PerfTestPlus

Gold Sponsor:

keytorc tested by

Silver Sponsor:

LOGO BUSINESS SOLUTIONS

Supportive Organizations:

STICKYMINDS.COM™ A TECHWELL COMMUNITY

MODERN analyst .COM

ea with ime esters

PROFESSIONAL TESTER

ISTQB International Software Testing Qualifications Board

TTB Turkish Testing Board


Every Tester
who reads Tea-time with Testers,
Recommends it to friends and colleagues .
What About You ?
Image : vernhart

in next issue

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

Core Team:

Dr.Meeta Prakash (Bangalore, India)


Dr. Meeta Prakash

Testing Puzzle & Online Collaboration:

Eusebiu Blindu (Brno , Czech Republic)

Shweta Daiv (Pune, India)


Eusebiu


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 **FREE** copy ,

Subscribe to our group at

Google™

Join our community on

facebook

Follow us on

www.teatimewithtesters.com