THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

MAY 2014 | YEAR 4 ISSUE IV


Musings over Tea-time
Anthology of T-Talks
T Ashok
DOWNLOAD YOUR FREE COPY
CLICK HERE
Celebrating THREE Years of Tea Time with Testers
ANNIVERSARY SPECIAL

Tea Time with Testers Special: For our Subscribers Only

11th International Software Testing Conference

STeP-IN SUMMIT 2014 / Testing NOW

Interaction » Insights » Opportunities

20% Discount Coupon Code: TTwT2014

TTwT brings its subscribers a special **20% Discount Coupon** to attend Asia Pacific's Largest Software Testing Conference: STeP-IN SUMMIT 2014 at any of the 3 Locations, Pune, Hyderabad & Bangalore. To avail the discounts quote **Coupon Code TTwT2014** when STeP-IN Forum calls you after your registration.

Conference Program | CLICK TO REGISTER

CONFERENCE PROGRAM

18, 19, 25 June, 2014 Pre-conference tutorial workshops (two parallel tracks) @ Pune, Hyderabad and Bangalore

26 - 27 June, 2014 Conference @ Bangalore

Over 900+ delegates from India and abroad attending STeP-IN SUMMIT 2014

Are You...?

For more details write to conferences@stepinforum.org or call Pinky on +91 - 80 - 4081 8888

Produced by:

STeP-IN Forum
www.stepinforum.org

www.stepinsummit.stepinforum.org

Hosted by:

QSIT The Knowledge Corporation
www.qsitglobal.com

ROI | REACH. OPPORTUNITY. IMPACT.

PER SCHOLAS 2014 CORPORATE DINNER

JUNE 12 | 6PM - 9PM

GRAND HYATT NEW YORK

109 EAST 42ND STREET @ GRAND CENTRAL TERMINAL

HONOREES

CORPORATE PARTNER OF THE YEAR **BARCLAYS**
PERSON OF THE YEAR **KEITH KLAIN**
INNOVATOR OF THE YEAR **DORAN JONES**

PRESENTING SPONSOR

BARCLAYS

SPONSORS

Bloomberg

CREATING IT FUTURES FOUNDATION

JPMORGAN CHASE & CO.

SPONSORSHIPS AVAILABLE

IMPACT 50K
OPPORTUNITY 30K
REACH 20K

BENEFITTING

PER SCHOLAS

FOR MORE INFORMATION
EVENTS@PERSCHOLAS.ORG
PERSCHOLAS.ORG/ROI2014

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:

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

Editorial

The State of Software Testing! Are we there yet?

You must be knowing about the 'State of Software Testing' survey that we conducted for year 2013. While the survey unveiled some interesting results there were still some interesting questions like:

- What do these results tell us about the current state and the future trends of the Testing Profession?
- What are the important trends defining the world of Testing today?
- How will the testing world look in the near and not so near future?
- What are the professional areas to develop in order to succeed as a tester in the future?

We approached testing experts Jerry Weinberg and Fiona Charles to review the survey results. You can find out what they answered in **webinar** that we conducted recently.

I would call this webinar a big success because of the overwhelming response and active participation from testers across the globe. #stateoftesting was trending on twitter for quite some days. It was great to know from Jerry and Fiona on different questions we asked them during the webinar and from their opinions; advice on different topics that we discussed. All in all their inputs and suggestions have definitely helped us gain better insights of things. And we plan to make this survey more useful and concrete for times to come.

Being admirer of craft of testing, a question that still makes me curious is "Are we there yet?" I think, not really. "Will we be there soon?" May be YES! Or it depends. It will be interesting to see what we get to know about State of Testing as we try finding it out by end of this year.

And before I take your leave, let me welcome Unmesh Gundecha who has joined our editorial team this month. Welcome to the family, Unmesh!

Yours Sincerely,

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

QuickLook

QASymphony Unveils 3 Tiers of Software Testing Tools

Scalable tools for QA people: qTest Project, qTest Pro, and qTest Enterprise, suit every size development project and organization

QASymphony (www.qasymphony.com), a leading provider of test management platforms for agile development teams, formally announced a major new update to its qTest test management platform. This update provides extensive integration capabilities through the formal release of its APIs, support for test automation tools such as Selenium and testNG, and features that help simplify management of large global teams.

Along with new capabilities, QASymphony is also offering the qTest test management solution as three separate editions. qTest Project, Pro, and Enterprise options provide testers with the flexibility they need to gain the benefits across different testing scenarios and Agile strategies. Available in the cloud or installed on-premise, the SaaS tools start as low as $1 per user per month.

While qTest Project is geared for small teams of less than 10 users working on single projects, qTest Pro is for small to mid-size teams looking to upscale and improve test management. qTest Enterprise, as the name implies, is designed to support many users on multiple projects and integrate with other enterprise-level apps.

QASymphony

"We've learned a few things from listening to our 8,000+ users, and the one clear message that stands out is that testers in various size teams want a testing tool tailor-made to fit the way they work. The 'one-size fits all' approach doesn't work," says Vu Lam, QASymphony co-founder. "In short, the qTest platform with various options to choose from enable teams to communicate better and test faster in agile environments."

The qTest platform provides a collaborative work environment for teams to manage requirements, design test cases, plan test execution, track defects, and generate status and quality-metrics reports. With support for both scripted and exploratory testing, qTest is the only platform that lets you add innovative exploratory to manual and automated test management, creating a consolidated solution that accelerates the testing process to keep pace with today's rapid agile software development.

Pricing and Support

Available today, qTest Project ($1/mo.) includes 24x7 email tech support. qTest Pro ($29 per user per month, billed annually) and qTest Enterprise ($49/mo.) additionally include 8x5 online and 8x5 telephone tech support services. Enterprise option includes automation support via TestNG and Selenium services (http://testng.org/doc/selenium.html).

qTesteXplorer, an add-on to qTest that automatically captures a tester's actions in both text steps and screen captures, is offered at $38/mo. Pay-as-you-go options are available.

qTest arrives fully featured with pre-built integration with popular development lifecycle tools and defect tracking systems with modules that include:

- Test Planning
- Test case management
- Defect tracking
- Built in workflows and collaboration
- Requirement management
- Test execution
- Dashboards and Reports

About QASymphony

QASymphony is a leading provider of testing solutions that fit the needs of development teams at any size. Whether you are running a small team and need basic testing help or you manage large global teams looking for better coordination and visibility, our test management and agile testing solutions can help. With HQ in Atlanta, GA, and operations in Ho Chi Minh City, Vietnam, QASymphony is a software company building tools that accelerate testing, helping you keep up with the pace of agile development. Empowering the QA testing teams for companies such as Good Technology, Silverpop, BetterCloud, and Zappos, QASymphony is a software-loving team, united by a common belief that good software can be delivered fast with high quality.

Website: www.qasymphony.com **Facebook:** www.facebook.com/qasymphony **Twitter:** www.twitter.com/qasymphony

Press Contacts: Victor Cruz (Principal, MediaPR) vcruz@mediapr.net

QASymphony

BARCLAYS, KEITH KLAIN, AND DORAN JONES TO BE HONORED AT ROI 2014

by – Jessica White

We are very excited to announce the third annual Per Scholas ROI Corporate Dinner taking place June 12, 2014, where we will honor:

Barclays, CORPORATE PARTNER OF THE YEAR

Keith Klain, PERSON OF THE YEAR

Doran Jones, INNOVATOR OF THE YEAR

Last year, Per Scholas hosted its most successful event in the organization's history when it honored Joe Squeri at the second annual ROI Corporate Dinner. The outcome of that evening extended far beyond that night resulting in three landmark achievements led by this year's honorees. We are thrilled to recognize the enormous new impacts that Barclays, Keith Klain, and Doran Jones brought to our work creating better futures for our students over the past year.

In a very short time, Barclays has become one of Per Scholas' most generous corporate partner – providing sizable funding, sending skilled volunteers on a monthly basis, creating employment opportunities for our graduates, and championing support from senior leaders. Most impressively, it played the leading role in bringing a new Software Testing Education Program (STEP) to Per Scholas last year.

While at Barclays last year, Keith Klain pioneered STEP and recruited other needed partners from across the software testing industry to help develop the curriculum, including Satisfice, QASymphony, UTest, Smartbear, Workroom Productions, DevelopSense, and the Association for Software Testing. As a direct result, more than 60 students have completed the STEP training, with graduates earning nearly 20% more than those from Per Scholas' other IT training courses.

Mr. Klain did not stop with STEP. He saw a much bigger opportunity — the creation of a fully operational software testing center that would create revenue, new jobs, and a much-needed service for the industry while employing hundreds of STEP graduates. The new Per Scholas Urban Development Center (UDC) was originally Keith's idea — but it needed one other partner to become a reality.

Doran Jones, an IT consulting firm, immediately recognized the potential to add to their services in software development, and signed up as Per Scholas' private-sector partner on the software testing job creation initiative. Managed by a world-class executive team with extensive experience in all aspects of software development and testing, Doran Jones are backed by a board with a strong track record in financial services and community investment. They will operate the UDC co-located at Per Scholas' headquarters in the South Bronx and provide software testing services to its clients, initially employing 150 Per Scholas STEP graduates over the next 18 months.

The initiative will also bring significant new income into the South Bronx neighborhood, one of the poorest in the U.S. Doran Jones is proving the win-win potential of corporate engagement with our work and our communities — and Per Scholas could not be more excited to celebrate this innovation and commitment.

The dinner is being co-sponsored by a large range of other Per Scholas corporate partners and allies, including **Bloomberg, JPMorgan Chase** and **the Creating IT Futures Foundation**, Per Scholas' chief partner in its national expansion. For more information and to attend as a sponsor or buy individual tickets, please visit **perscholas.org/roi2014**.

A million dollar smile ?

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

*Adverts starting from $100 USD | *Conditions Apply

Photo credits: Amir Eltahan

Through her article series in Tea-time with Testers, I got to know more about Bernice Niel Ruhland.

What has impressed me more than anything else about Bernice; is her passion and seriousness towards driving the change in a way testing is done.

Bernice's tireless contribution to community in the form of blogs she writes and articles she has written for multiple publications till date is evidence of her desire to create difference. I feel amazed at times when she pro-actively looks out for solutions to testing problems her teams face and for things that could help her teams perform better. I have seen very less managers working at her level, going out of the way to help their teams acquire new skills, to get them on-boarded with new test ideas and to keep them motivated in all possible ways. What more could one hard working junior tester ask for? A leader like Bernice, for sure.

In my opinion, Bernice is true example of leading by example and of person who walks the talk.

What makes Bernice so passionate about craft of testing? How does she manage all things she does? What is her opinion about different things around Testing?

I got a chance to discuss things with Bernice and I will let you find out what we all discussed about.

Naturally, in this exclusive interview….

- Lalitkumar Bhamare

Over a Cup of Tea with Bernice Niel Ruhland

Bernice Niel Ruhland is quite familiar name in software testing community. We are curious to know about your testing journey. Was it by choice or by chance?

I started my testing career by attending a technical college for computer programming where we were taught how to code, flowchart solutions, and test. For about 12-years I was a programmer in various business disciplines including Finance, Human Resources, and Securities. Our responsibilities included: meeting with the business users to write requirements; code the reports or solutions; and conduct the testing. And of course I loved the testing part! My responsibilities transitioned to a project leadership role during the Y2K scare to ensure that production programs were correctly converted and tested. At that time I completed my Bachelors in Management and started my management career by forming a Software Development department and then a Software Testing department. To further my career I completed my Masters in Strategic Management, which led to a position with responsibility for identifying and implementing company-wide solutions. My current position consists of the strategic oversight of a Software Testing department and company-wide quality programs. In reviewing my career, testing has always been an important component and I am a born tester.

What testing related activities do you conduct for your teams, which are outside of your regular test deliverables?

Continuous learning is important for every tester from both an individual perspective and group effort. I am a big fan of journal clubs. They are a wonderful way for a group of testers to expand their knowledge on how other testers are approaching similar problems. Sometimes I facilitate these sessions or a group of testers may get together and rotate the responsibility. It is best when more than one person is selecting the article for discussion. It is great to see the topics testers select that they find interesting. I recommend for teams to consider expanding articles to include short webinars to keep it fresh. Testing challenges are also a great way to learn and can be a fun activity during a lunch break.

Has any of them helped you as a Test Manager? How? And would you like to share some tips for other Test Managers who would like to try them?

It is important to encourage Testers to learn about different test techniques and approaches to ensure a wide-range of knowledge and skills. Plus connecting with other Testers to understand how they approach similar problems is important. When encountering a testing problem you may need to go to your testing toolbox or contact someone in the Testing community to discuss potential solutions. As an example, I read a lot about mind mapping and Kanban boards before having an opportunity to use them. When the opportunity arose, I already knew about them and with a bit of experimentation we were up and running. My recommendation to Test Managers is to stay on top of testing trends; determine when deeper training is required; and consider the future of your department.

How do you balance things between test management and a people management part of your role? Are there any experiences / experiments that you would like to share?

I believe a manager's role is to empower employees to take ownership of their job and to provide continual learning opportunities to expand their knowledge and skills. A manager must care about their employees to ensure they have the best work environment and to be a champion for them. But perhaps we view test management and people management collectively instead of separately. As managers we need to work with our team to discuss activities such as testing risks, scope, and approaches. We can enrich their jobs through pairing Testers to provide cross training. Understanding Testers strengths and career aspirations can help with identifying opportunities. A strong Test department, in my opinion and experience, is a team effort and not a silo.

.... CONTINUED ON PAGE 52

Tea & Testing with Jerry Weinberg

The lights at the end of the tunnel

A long auto tunnel through the mountains above Lake Geneva has just been completed. Just before the opening, the chief engineer remembers she has forgotten to warn motorists to turn on their lights before entering the tunnel. Even though the tunnel is well illuminated, the motorists must be prepared to prevent a catastrophe in the event of a power failure—a plausible eventuality in the mountains.

A sign is made saying:

WARNING: TUNNEL AHEAD
PLEASE TURN YOUR HEADLIGHTS ON.

The tunnel, with the sign well ahead of the entrance, is opened on schedule, and everyone relaxes, now that the problem is solved. About 400 meters past the Eastern end of the tunnel stands the world's most scenic rest stop, with a sweeping view from high above the lake.

Hundreds of tourists stop there each day to enjoy the view, perform important bodily functions, and perhaps partake of a small but tasty picnic.

Figure 1. Tourists in the Alps taking a picnic after passing the tunnel.

And every day, ten or more of those hundreds return to their cars, refreshed in body and soul, only to find a dead battery from having left their lights on! The gendarmes are tying up most of their resources getting them started or hauling them away. Tourists are complaining and swearing to tell their friends not to visit Switzerland.

As usual, we ask you to pause and ask yourself:

WHOSE PROBLEM IS IT?

(a) the drivers
(b) the passengers (if any)
(c) the chief engineer
(d) the gendarmes
(e) the president of the canton
(f) the automobile clubs
(g) none of the above
(h) all of the above

The strong tendency in this type of problem—with an explicit "designer" or "engineer"—is to consider her problem. Not only do the drivers in this case consider it the engineer's problem, but the engineer probably does too. It's a common impression among architects, engineers, and other designers that they must take care of everything.

In this instance, the engineer considered various solutions she could impose upon the drivers and their passengers.

(1) She could put a sign at the end of the tunnel saying, TURN OFF YOUR LIGHTS but then people would turn off their lights at night...

(2) She could ignore the situation and let people...No, that was already happening, and the government officials think the engineer has done a lousy job.

(3) She could put a battery-charging station at the scenic overlook. But that would be expensive to maintain, and would make people even more furious if it didn't work.

(4) She could give the recharging station franchise to a private firm. But that would commercialize the overlook and be unacceptable to the government and the tourists.

(5) She could put a more explicit sign at the end of the tunnel.

The engineer felt intuitively there should be some way to write a more explicit sign. She worked on several alternatives and eventually came up with a masterpiece of Swiss precision:

IF IT IS DAYLIGHT, AND IF YOUR LIGHTS ARE ON, TURN OFF YOUR LIGHTS;

IF IT IS DARK, AND IF YOUR LIGHTS ARE OFF, TURN YOUR LIGHTS ON;

IF IT IS DAYLIGHT, AND IF YOUR LIGHTS ARE OFF, LEAVE YOUR LIGHTS OFF;

IF IT IS DARK, AND IF YOUR LIGHTS ARE ON, LEAVE YOUR LIGHTS ON.

By the time anybody had finished reading this sign (in three languages), his car would be over the guardrail and gurgling to the bottom of the lake, which would not be an acceptable solution at all. Besides, what about funerals? There must be a better way!

Instead of all this complication, the chief engineer took the approach of "It's THEIR problem"—but it was her problem to assist them. She assumed the drivers had a strong motivation to solve the problem, but they might need a little reminding. She also assumed the drivers—if they were to be: licensed at all—couldn't be complete dummies. All they needed was a sign at the end of the tunnel:

If they weren't smart enough to deal with that, dead batteries were the least of their problems. This sign eliminated the problem, and the message was short enough to be put on the sign in several languages. The engineer always remembered her lesson from this situation:

IF PEOPLE REALLY HAVE THEIR LIGHTS ON, A LITTLE REMINDER MAY BE MORE EFFECTIVE THAN YOUR COMPLICATED SOLUTION.

Are your lights on?

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 .

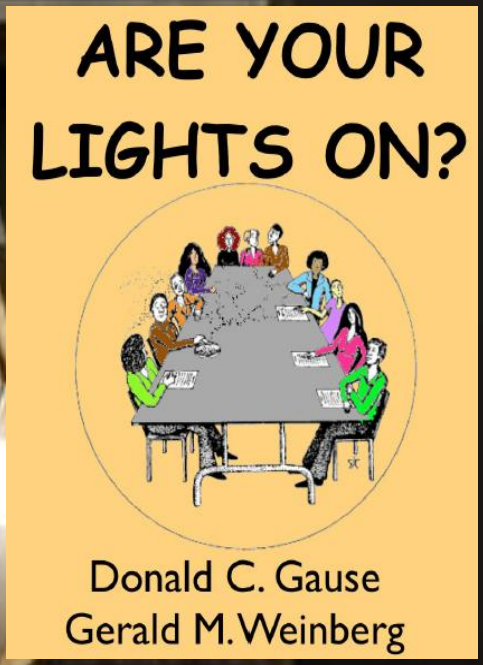
Gerald can be reached at hardpretzel@earthlink.net or on twitter @JerryWeinberg

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.

Its sample can be read online here.

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



TTWT Rating: ★★★★★

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.

The Tester/Hacker Mindset

- by **Lorinda Brandon**

I talk a lot about APIs –sometimes I talk about testing them, sometimes I talk about designing them, and often I talk about marketing them. As I prepared for my presentation at last month's API Strategy & Practice Conference I found myself combining all of that into one discussion for the Security & Testing track.

When we discuss marketing APIs, we often talk about developer outreach and making your API so easy to adopt that everyone wants to use it. The focus is on opportunity and innovation. What we seldom acknowledge is that this is the exact thing we guard against when we put together our security plans. When we wear our security hats, we worry about the kinds of innovative misbehavior that might occur.

In essence, marketing APIs and developing API security policies are really about the same thing – innovation.

On the marketing side, we want to encourage developers to innovate and use our APIs constructively. But from the security perspective, what really happens when we put an API out there into the public sector is that people often find innovative ways to use our APIs and/or the data destructively. It's kind of the yin-yang of API innovation. Either way, it's about trying to figure out what sort of creative, crazy, inventive things are people going to do with your data that you didn't anticipate and that you didn't think of yourself.

Security protocols are only part of the story – what we sometimes fail to remember is that people can walk right in your front door and still steal the silverware. Some of the most prominent examples in the news lately were victims of just that – hackers using the API as it was designed but finding that the design itself had holes they could exploit.

One example is Snapchat, whose API allowed for some creative misbehavior that resulted in people being able to connect phone numbers with all the users in the Snapchat ecosystem. Additionally, they could use the API to easily create thousands of bogus Snapchat users. Unfortunately for Snapchat, they ignored the original warnings from the hackers themselves and were eventually publicly embarrassed when the hackers went public.

(There's a whole separate discussion to have about the legalities of hackers publishing exploit details publicly – something our legal system needs to catch up with if we're going to protect both companies and their users.)

Likewise, Tinder also had a famous exploit where their API passed enough information about their users' locations that clever developers could physically locate people. Even worse, they reacted quickly to the first report of this vulnerability… by releasing a fix that didn't solve the problem. This vulnerability is one of the rare instances where people were theoretically put at physical risk, allowing a diabolical

stalker/hacker to find unwitting Tinder users without their knowledge. Luckily, that didn't happen… but it's all about possibilities, right?

The Tester/Hacker Mindset

The key to preventing these vulnerabilities and security issues lies in having the right mindset. Typically, I think of both developers and operations people as having the mentality of builder, checker, and analyzer. When it comes to quality, they'll generally ask questions like:

- Does this work the way I thought it would?
- If I inject an error into it, does it return the right message?
- If I put rate limits on my API are those honored?
- If I go above rate limits, am I blocked?

On the other hand, the tester mindset is closely aligned with how hackers think. Skilled hackers and testers approach a product (including your API) from a different vantage point – once they get past checking, they start testing:

- Now that I know the boundaries, how strong are the walls marking those boundaries?
- What can I do besides the obvious documented activities?
- What if I try this? Or this? Or this?
- Is there a way to go down a different path than the product obviously wants me take?

They do not feel obligated to use your API in the way you intended. They're the ones who are taking all of that critical thinking, those "out of the box" ideas, and really testing the limits of the data that you're delivering. They're trying to poke holes and take advantage of that seemingly innocuous vulnerability in your API.

A new approach to testing is imperative in the new world of software development. We're already moving away from the waterfall, process-heavy, metrics-heavy testing of the past. But we still need to think about new strategies to keep up with the API boom and the Internet of Things, both of which are changing the landscape of technology in a major way.

So, how can you find the vulnerabilities your test scripts can't uncover?

If you really want to get the ultimate coverage when testing your API, embrace the hacker mindset and bring hackers onto your team. Work *with* them, rather than against them. Listen to them and learn how they think about that data you're delivering. If you bring them into your process early enough, before you've delivered your API to market, you can deploy with a greater confidence.

Another strategy is to put the "hack" back into hackathons. Sure, we've been talking about the benefits of hosting hackathons for years. But the hackathons that most organizations set up are maker events. Those hackathons focus on getting as many developers as possible building as many apps as possible, using your API. Exactly the thing security folks worry about.

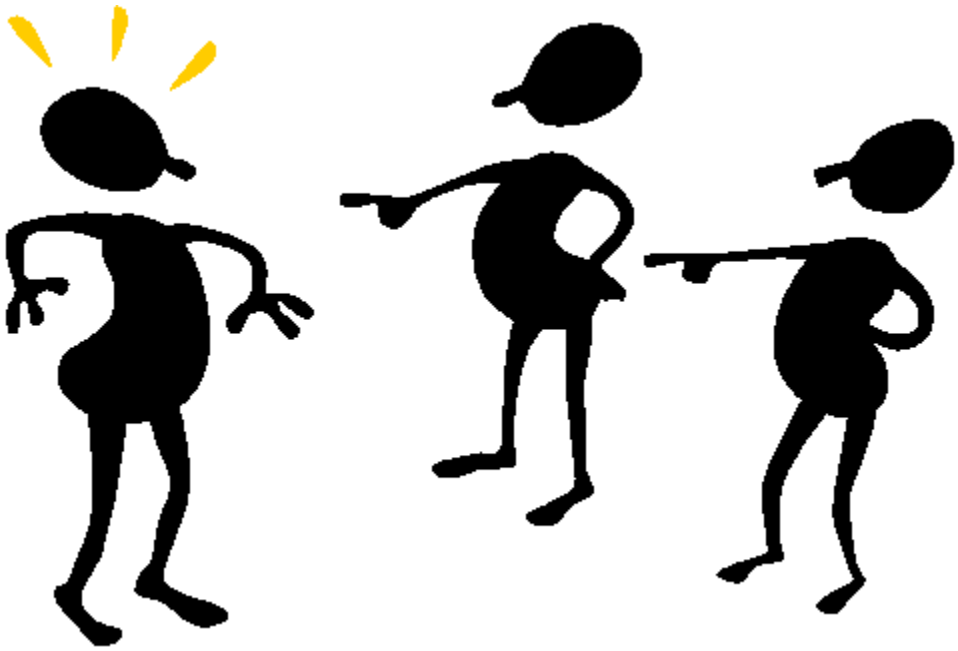
So, what if we hosted hackathons where we really encourage hackers to get creative in malicious ways? Find the vulnerabilities before you go to market by using the same strategies you use when marketing your API.

We've got to start realizing that security is not just about protocols, gateways, and rate limits. We need to embrace the innovative misbehavior of hackers as one of the best ways to test our APIs and the data we deliver.


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

Blaming the Tester

- by Adam Knight



It has been my unfortunate experience more than once to have to defend my testing approach to customers. On each occasion has been deemed necessary in the light of an issue that the customer has encountered in the production use of a piece of software that I have been responsible for testing. I'm open to admitting when an issue that should have been detected was not. What has been particularly frustrating for me in these situations is when the presence of the issue in question, or at least the risk of it, had already been detected and raised by the testing team...

The Dreaded Document

I have a document. I'm not proud of it. It describes details of the rigour of our testing approach in terms that our customers are comfortable with. It talks about the number of tests we have in our regression packs, how many test data sets we have and how often these are run. The reason that the document exists is that, on the rare occasion that a customer encounters a significant problem with our software, a stock reaction seems to be to question our testing. My team and I created this document in discussion with the product management team as a means to explain and justify our testing approach, should this situation arise.

The really interesting element in exchanges of this nature is that no customer has ever questioned any other aspects of our development approach, irrespective of how much impact they may have on the overall software quality.

- They do not question our coding standards
- They do not question our requirements gathering and validation techniques
- They do not question our levels of accepted risk in the business

- They do not question our list of known issues or backlog items
- They do not question the understood testing limits of the system

Instead they question the testing

This response is not always limited to external customers. In previous roles I've even had people from other internal departments questioning how bugs had made their way into the released software. I've sat in review meetings listening to how individuals '... thought we had tested this software' and how they wanted to find out how an issue 'got through testing'. Luckily in my current company this has not been the case, however I have had to face similar questions from external customers, hence the document.

A sacrificial anode

The behaviour of the business when faced with questions from customers over testing has typically been to defend the testing approach. Whilst this is reassuring and a good vote of confidence in our work, it is also very interesting. It seems that there is a preference for maintaining the focus on testing, rather than admitting that there could be other areas of the business at fault. Product owners would apparently rather admit a failure in testing than establishing a root cause elsewhere. Whilst frustrating from a testing perspective, I think that on closer examination there are explainable, if not good, reasons for this reluctance.

- **The perception of testing in the industry -** Whilst increasing numbers of testers are enjoying more integrated roles within development teams, the most commonly encountered perception of testing within large organisations is still of a separate testing function, which is seen as less critical to software development than product management, analysis and programming. As a consequence of this I believe that it is deemed more acceptable to admit a failure in testing than other functions of the development process which are seen as more fundamental. A reasonable conclusion then is that, if we don't want testers to receive blame for bugs in the products then we need to integrate more closely with the development process. See here for a series of posts I wrote last year on more reasons why this is a good idea.

- **Reluctance to admit own mistakes -** Often the individuals asked to explain the presence of an issue that was identified by the testing were the ones responsible for making the decisions not to follow up on that issue. In defending their own position it is easy to use a mistake in testing as a 'sacrificial anode' in removing attention from risk decisions that they have made. This is not purely a selfish approach. Customer perception is likely to be more heavily impacted by an exposed problem in the decision making process than one in the testing, as a result of the phenomenon described in the previous point. It therefore makes sense to sacrifice some confidence in testing at the expense of admitting that a problem arose due to the conscious taking of a risk.

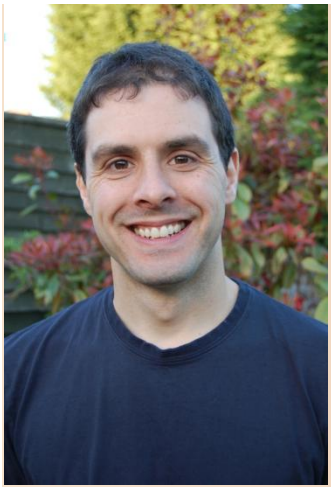
- **"Last one to see the victim" effect -** A principle in murder investigation is that the last person to see the victim alive is the likeliest culprit. The same phenomenon applies to testing. We're typically the last people to work on the software prior to release, and therefore the first function to blame when things go wrong. This is understandable and something we're probably always going to have to live with, however again the more integrated the testing and programming functions are into a unified development team, the less likely we are to see testing as the ones who shut the door on the software on its way out.

Our own worst enemy

Given the number of testers that I interact with and follow through various channels, I get a very high level of exposure to any public problems with IT systems. It seems that testers love a good bug in other people's software. What I find rather disappointing is that, when testers choose to share news of a public IT failure, they will often bemoan the lack of appropriate testing that would have found the issue. I'm sure we all fall into this mindset, I know that I do. Whenever I perceive a problem with a software system, either first hand or via news reports, I convince myself that it would never have happened in a system that I tested. This is almost certainly not the case, and demonstrates a really unhealthy attitude. By adopting this stance all we are doing is reinforcing the idea that it is the responsibility of the tester to find all of the bugs in the software. How do we know that the organisation in question hasn't employed capable testers who have fully appraised the managers of the risks of such issues, and the decision has been made to ship anyway? Or that the testers recommended that the area was tested and a budgetary constraint prevented them from performing that testing. Or simply could it be that the problem in question was very hard to find and even excellent testing failed to uncover it? We are quick to contradict the managers who have unrealistic expectations of perfect software, claiming the infinite combinations of functions in even the most simple systems, yet we seem to have the lowest tolerance of failure in systems that are not our own.

Change begins at home and if we're to change the 'blame the tester' game then we need to start within our community. Next time you see news of a data loss or security breach, don't jump to blaming the thoroughness, or even absence, of testing by that organisation. Instead question the development process as a whole, including all relevant functions and decision making processes. Maybe if we start to do this then others will follow, and the first response to issues won't be to blame the tester.

Back To Index



Adam Knight has been testing data storage and analysis software for 10 years. He is an enthusiastic exponent of exploratory testing approaches backed by discerning use of automation and is a great believer in creating multi-skilled testing teams based on rich and unique individual skill sets. At his current employer, RainStor, Adam has overseen the testing and technical support of a large scale data storage system from its initial release through to successful adoption in some of the largest telecommunication and financial services companies in the world. His work at RainStor was used as a case study in the successful 2011 book "Specification By Example". Adam has spoken at numerous testing events including Agile Testing Days, EuroStar and TestBash. He is on twitter as @adampknight and writes a blog at www.a-sisyphean-task.com.

Tea, Testing and

QASymphony

Discover how qTest can accelerate your testing

The race to modernize software development is all about cutting the cycle time between conception and production delivery. It is all about breaking down superfluous processes and inefficient bottlenecks to speed up development without sacrificing quality. A lot of time and energy has been expended on honing the development and deployment of new software, but there's another area that's ripe for an overhaul -- testing.

QASymphony's **qTest** platform provides a consolidated solution for the modern test team that's capable of managing the entire software testing life cycle and beyond. This is a cloud-based collaborative work environment that empowers teams with the ability to plan, design, and execute test cases; track and manage the defects workflow; and generate status and quality-metrics reports. It's designed to dovetail with modern development practices and systems, with support for scripted or exploratory testing. It is also built using open architecture which integrates seamlessly with 3rd party requirements management, defect tracking systems, and test automation tools.

Save testing time with qTest

There are many benefits to be gained from adopting qTest. As a cloud-based SaaS solution it is easily scalable, accessible, and suitable for a distributed workforce. It can also boost communication as a centralized repository for all testing activities. However, in this article we are focusing on how qTest can help accelerate your testing efforts and make your testing team more productive.

Here are just a few ways the platform can save you time:

Defining and Documenting Test Cases

With qTest you can dive straight into exploratory testing and your early test sessions can actually form the basis of your test cases. There's no need to waste time working through the software and documenting the steps separately because qTest can automatically capture your every move and convert into well documented test cases. You can edit these steps and link them directly to requirements and then assign and schedule them as part of your test plan.

Execution of Test Cycles

qTest features TestPad, a miniature version of the test execution module that can display next to the target application and provide a compact view of the steps that need to be executed. Testers can also mark results, update test cases, and create defects. As testing progresses, qTest will automatically track execution results, update test cases, and load subsequent test cases into the TestPad.

Exploratory Testing

If you really want to speed up your testing, adopt exploratory testing as part of your testing methodology. You can skip test case writing, get into actual testing as soon as builds are available, and be able to get feedback to the developers immediately. The qTesteXplorer tool let you implement this lean testing technique while still providing testing documentation and metrics for tracking and reporting purposes.

Tracking Test Execution

qTest provides a workflow with a clear record of what test cases have been executed for any specific build -- which test cases failed, what defects were linked to those test sessions, and what work remains to be done. There's a concise overview available that ensures work is never duplicated or missed, and testers always know what needs to be worked next.

Managing defects

Whether you are using Jira, Rally, VersionOne, Fogbugz, Bugzilla, or qTest,to track your defects, your existing defect tracking system can be integrated with qTest with simple configuration options. There's no need for a second screen or window where testers manually enter defects. There's no need to manually record the steps for reproduction or test environment information.

With qTest the steps and environment information are automatically recorded and the tester can edit as needed and enter the defect straight into the system. All of the pertinent details are automatically included when the defect is created, so there's no need to break away from the test session in progress; this speeds up defect creation drastically, as well as ensuring that defects are fully documented -- leaving no room for human error.

Generating business reports

Extracting data and using it as the basis for a report can be challenging for testers as many test management systems do a poor job when it comes to producing meaningful reports. With qTest all of the data about every test case, test session, and defect on every build of the software is instantly accessible. You can get a snapshot of where the current build is, you can discover the average velocity of an individual tester, and you can identify trends and potential issues. The ability to automatically generate business reports doesn't just save time, it could empower you to suggest changes that have a major impact on overall productivity or uncover issues that require attention.

Back To Index

Time to try qTest

For modern test teams distributed across the globe and working round the clock, a cloud-based SaaS platform is obviously the way forward, but on-premise installation is also available. This is a flexible solution that's designed to meet the needs of a diverse range of test teams.

How much time you actually save depends on your project and your test team set up, but as a consolidated test management platform, qTest clearly maximizes efficiency and reduces the risk of data loss and human error.

Simplifying the administration around testing also frees up your testers to do what you hired them to do in the first place – test the software, find defects, and improve the final quality of your product.

The **qTest platform** is a tailor-made test management solution for your business at an off-the-rack price.

Kaushal Amin is Chief Technology Officer for **KMS Technology, (www.kms-technology.com)** a software development and IT services firm based in Atlanta, GA and Ho Chi Minh City, Vietnam.

He was previously VP of Technology at LexisNexis and a software engineer at Intel and IBM. You may reach him at kaushalamin@kms-technology.com.

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

Browser tools – The untapped resource for Mobile Web Testing

- by Raj Subramanian

The mobile web has changed by leaps and bounds in the past few years. It has evolved so rapidly that, vendors and other website owners are finding it hard to keep up with it. Based on a recent study, about 28% of the internet usage comes from mobile phones and it is projected to take over desktop usage by 2014. With this being the case, users expect mobile friendly versions of websites as they access it on the go when they are travelling, at work, at home and doing normal day to day activities. Would you go to a desktop website just because a vendor does not have the mobile version of it? NO, you immediately lose interest and find other means to get the required information by going to your competitor.

So, as you observed from the above details, having mobile web applications has become almost mandatory and testing them has become even more challenging. You may ask "Why"? This is because there are different devices, form factors, screen resolutions (over 200 unique resolutions), different browsers, different rendering engines and countless permutations.

Thus, it becomes all the more important to think about different ways and approaches for testing the mobile web. One such approach is the use of *"Browser tools".* So let's start with the basics: what are browser tools?

Introduction to Browser tools

Browser tools are inbuilt consoles which are part of your browser and helps to inspect client side JavaScript, CSS, HTML5 code. Click on "F12" on Chrome or IE browser and you will see all the details and errors on the page.

Using these consoles you can get different information such as:

- Which files are being loaded when a page is accessed. This includes JavaScript, CSS, HTML5 files.
- What is the page load times, size, response codes etc.
- How much memory is consumed by the pages
- Details about the cookie data of the pages

And much more….

Most importantly it helps to catch errors due to JavaScript, CSS and HTML 5. These errors occur due to missing functions, missing files, errors in the code, missing images etc.

Types of information you can get from browser tools

Let's open any sports website, say www.nba.com. Once the page opens up click on "F12" on your Chrome or IE Browser (I am using Chrome for this example), you will see the below console opening up

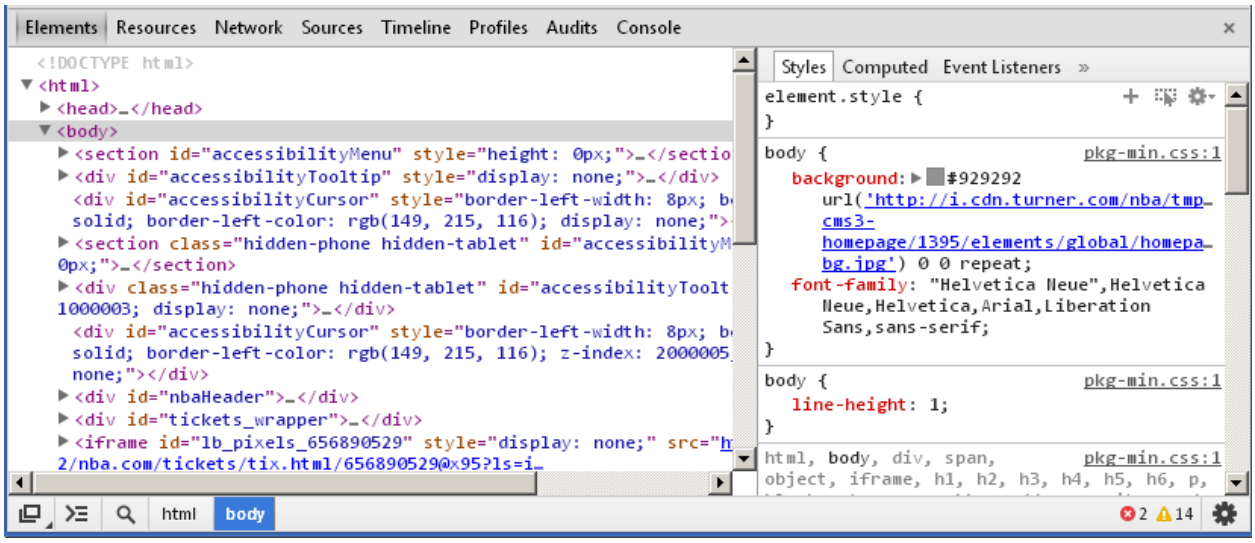


Figure 1: Chrome browser console

The console has different tabs. For the purpose of simplicity, I will go over the 3 main tabs that would be useful for getting feedback about web pages and gives valuable information to testers.

"Elements" tab – This tab shows all the client side JavaScript, CSS and HTML5 elements of the page. The tester can even vary the CSS on the right pane in Figure 1 and see how different changes affect the page.

Apart from this, you get valuable information regarding any JavaScript errors on the page. In Figure 2 below, notice the JavaScript errors and warnings in the page as indicated by the red box. On clicking on the errors, you get information about what error occurs, in which file and where as indicated in Figure 3.

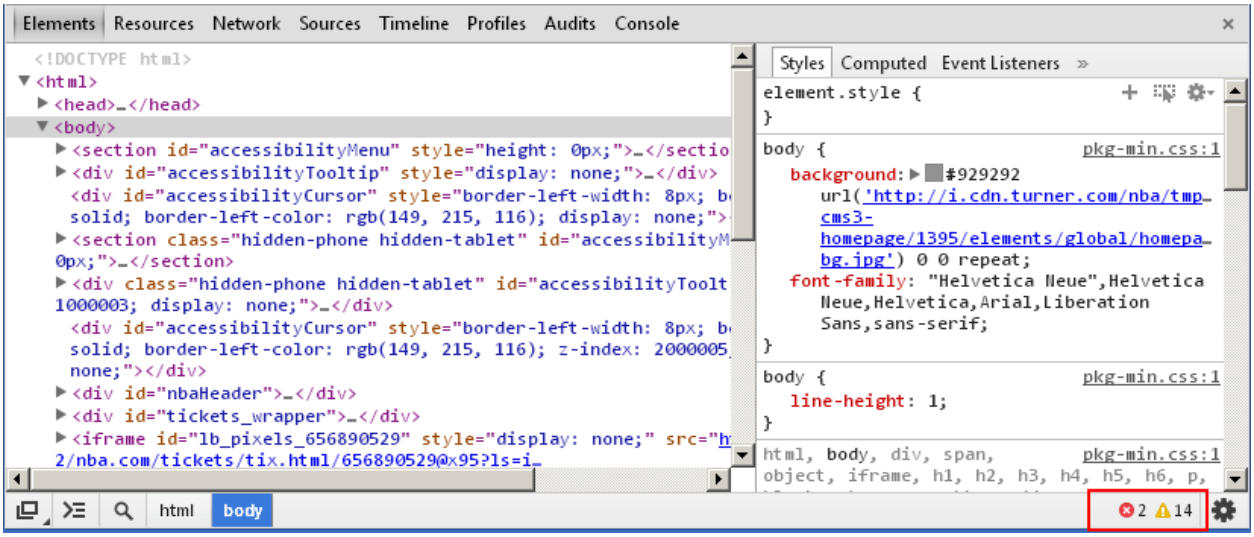


Figure 2: Chrome browser console gives the user information about JavaScript errors and warnings

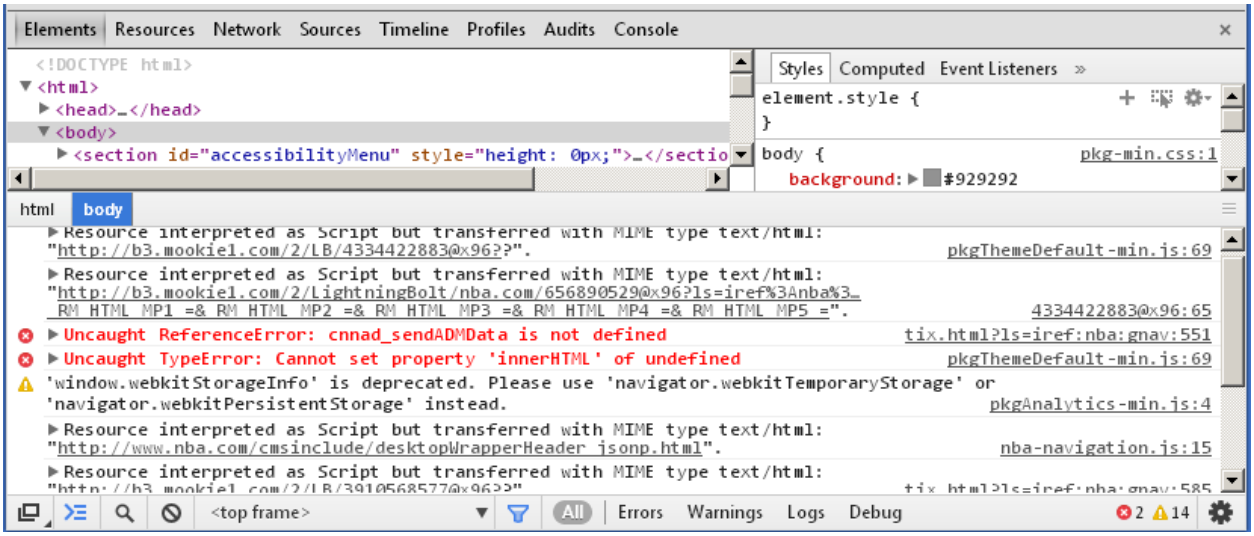


Figure 3: On double clicking the JavaScript errors and warning, a window opens up showing all the errors present in the page including information about which file and what line number the error is occurring

"Resources" tab – Clicking on this tab, you will be able to all the relevant details regarding the frames used, local storage, the sessions created, cookies and cached data (Shown in Figure 4 below). This becomes especially important when you are testing banking, insurance or other applications where sessions are stored, caching takes place and cookie data is created.

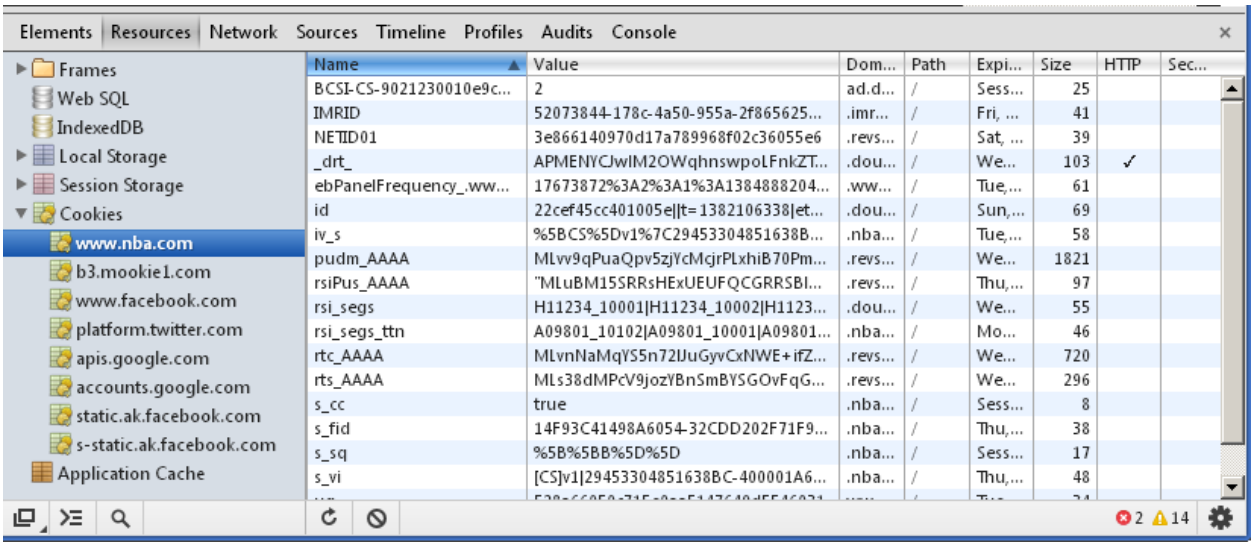


Figure 4: Resources tab in Chrome browser console

"Network" tab - This tab give valuable information regarding the page performance which is often a bottleneck in many web applications. This may be due to large files being loaded, unnecessary server calls being made or presence of large unused JavaScript files which affects the page load times.

Figure 5: The Network tab showing important page performance information

In the above figure, you can see that the "Network" tab shows information about different files being loaded when the page is displayed, in terms of what HTTP Methods are being used GET/POST, what is the status code of a particular file being loaded, the size of the file, the time it took for the file to load etc. This information is very valuable when investigating what files in a page are leading to slow response time of the page.

Why and how are the browser tools effective for testers?

The above 3 tabs serve a valuable source of information for testing mobile websites and helps to do quick smoke tests even before testing the websites on the mobile device. Think about this scenario - You are testing the "Login" page of a mobile website. When the developer pushed the code to QA, he/she accidently deleted the function which handles the "Submit" functionality.

Scenario 1 – Without using browser tools -

- Code is pushed to QA
- The testers take a couple of mobile devices and open the Login page.
- They give different username and password combination and click on "Submit"
- Nothing happens
- They try different username and password combinations (valid and invalid) click on "Submit"
- Still nothing happens
- The testers verify the page on tablets and other mobile devices and still see the same result
- They then create a bug report and assign the defect to the concerned developer
- The developer then takes a look at it and notices the missing JavaScript function which handles the "Submit" scenario
- He fixes it and pushed the new code back in QA
- The testers open up the page and verifies the issue on different mobile devices
- The testers then close the defect with the necessary comments

Scenario 2- Using browser tools

- Code is pushed to QA
- The testers open up the mobile version of the login page on the desktop and click on F12
- They immediately notice JavaScript errors on the page as shown by the console
- On clicking on the error, they are navigated to the file which has the error and the line at which the function is missing
- They immediately tell the developer where there is a problem and asks him to add the function
- Testers create a bug report and assign the defect
- The developer adds the missing function
- He pushes it back to QA

- The tester verify the fix by clicking F12 on the browser and making sure there are no JavaScript errors and also verifies the login scenario using the "Submit" button
- The testers then close the defect with the necessary comments

Scenario 1 and 2 above may look similar but there is a major time saving factor here. Verifying a page in different mobile devices, phablets and tablets consumes a lot of time. Instead, you first do a smoke test using the browser tools verify there are no errors and then test the page on different devices. This helps to catch defects early.

An even more time saving factor would be, if developers could use the browser tools for smoke tests even before pushing the code to QA. Then, you could eliminate both the scenarios above and save valuable testing time.

So, to summarize, browser tools help to-

- Catch defects early
- Saves valuable testing time
- Points out where the problems in client-side code exactly is and help developers to fix the issue quickly
- Serves as a smoke test to make sure major functionalities are not broken even before starting testing on multiple devices.

If you have some other interesting ways of finding defects or feel I got something wrong here, feel free to add them in the comments. I would be happy to discuss. Happy reading!

Raj is a passionate tester, who has previous experience working as a developer and moved to testing to focus on his passion. He graduated from Rochester Institute of Technology with a Masters in Software Engineering, worked as a developer for a payroll processing company and currently works as a test engineer for a major insurance company with specific focus on mobile testing. He has been pretty active in terms of learning and contributing to the testing society by speaking at conferences, writing articles, blogging and being directly involved in various testing related activities. He currently lives in Cleveland, Ohio.

e-mail -> raj@rajsubra.com
Website -> www.rajsubra.com
Blog link -> http://www.rajsubra.com/blog/
Twitter handle -> @epsilon11

Back To Index

SMART -

A Comprehensive Framework for Test Automation of Web & Mobile Applications Using Open Source Technologies

- by Anmol Bagga

Applications can be of different forms, it can be desktop based application, web application or mobile application. According to the current market scenario people are moving more and more to the smart devices. The trend has forced companies to develop applications across all the platforms to survive in the competitive IT Industry. As a QA Engineer, we must ensure that the application is supported across all the browsers, platforms and supported devices. Given the number of platforms and devices compatibility testing becomes a tedious task. Testing involves extra time, resources and cost. Moreover we need different tools to do automation testing on different platforms. Also the platforms are different but the product is same. The business actions, features and functionality will be similar across the platforms.

We also need functional testers who are not only good in their domain and testing but they should be also good in writing test cases which the test automation tool understands. We don't have such time luxury in an actual project environment to provide training to team members. Therefore a common language is needed which should be used as blueprint across the team.

At QA Infotech through our extensive research and development we have created a comprehensive and unified framework "***SMART***" to overcome such hurdles during test automation of complex applications which is available across varied platforms and plethora of devices.

The framework as its base uses open source technologies and is driven by behaviour driven development approach for test management.

Through our presentation we will discuss what are the different types of application, the challenges in testing them, current automation solutions and their challenges, our framework concept and design, how to implement your test cases on our framework, framework in action, Benefits and summary.

Different Forms of Application

Application can exist in different forms. We can have a desktop based application that can be installed and used by users on their laptops as well as desktops. For example calculator, ms office suite etc. Then we can have web applications that can be accessed through browsers. A user can access web applications through their desktop, laptops and smart devices like tablets, mobile phones. For example www.facebook.com. Then we can have mobile applications that can be installed and used on smart devices like mobile phones and tablets natively. For example whatsapp, we-chat etc.

(Figure 1)

Challenges in Testing

As an independent offshore testing company we at QA InfoTech make sure that the application is compatible across various forms of platforms, browsers and devices. Given the number of browsers, devices and platforms, compatibility testing becomes a tedious task. It involves extra time , resources and cost.

(Figure 2)

Current Scenario and Approach

There are various forms of application available to us. There is plethora of tools to automate them. For instance, when we talk about web applications we have selenium the most popular, free and open source, cross browser compatible tool. Then we have got QTP which is paid, licensed but supports desktop as well as web applications. and when we talk about automation tools for mobile there are tools like calabash and robotium that supports native android as well as iOS applications.

Although we have got various tools for test automation, we still have one big problem and the problem is that there is no single tool that works across all the platform and devices.

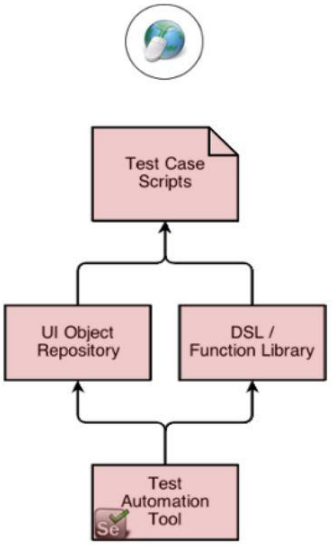
(Figure 3)

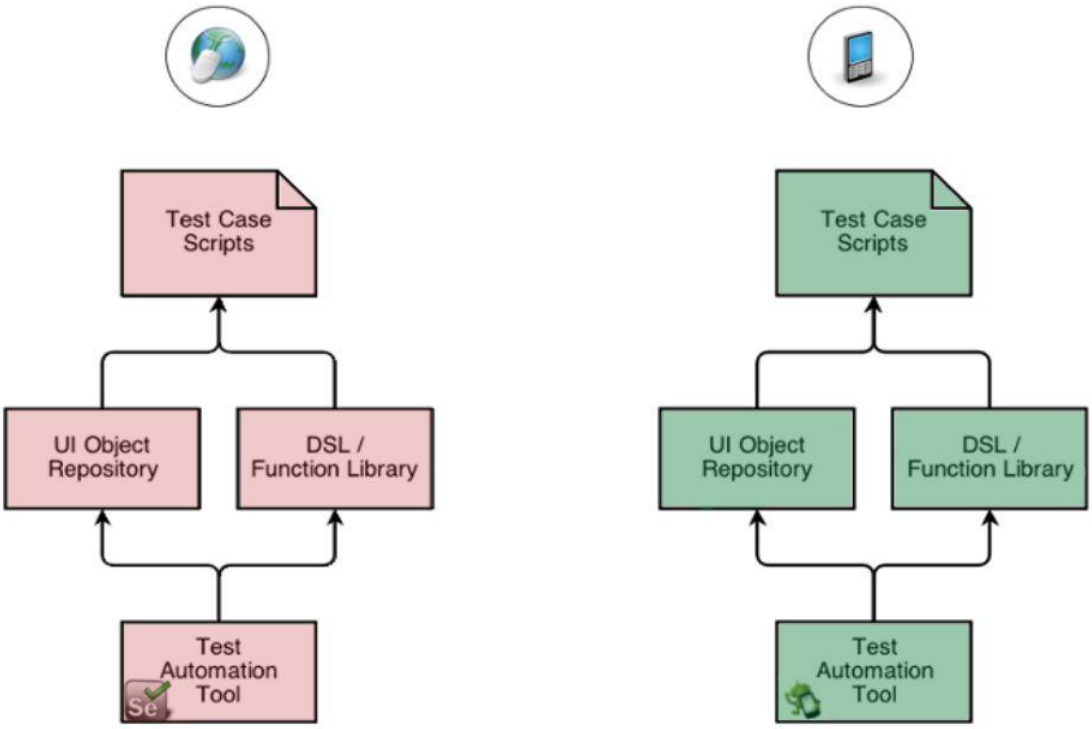
Selenium can handle web applications but it doesn't work on mobile applications. QTP can't work in linux, mac and on safari. Same is with mobile automation tools like calabash and robotium. They can handle mobile applications but they can't support applications running inside web browsers i.e. web applications.

Suppose a bunch of web developers are creating a web application and as a member of testing team you need to ensure that there is enough test automation coverage. So you first of all select the tool and then start creating your automation framework on top of the tool. Select your keywords or those actions that can be used again and again (also known as Domain Specific Language of the product), capture the page elements and page objects and put them in your object repository. Now an engineer writes test cases or test scripts and finally executes them in order to see that the application is working according to the specification or not.

Now after some time the web developers also decide that they need a mobile version of the application as well. They create an android as well as iOS version. But the automation framework which was working fine on the web application can't be used to automate the mobile version of the application. As the automation tool doesn't support mobile applications. So we have to create another framework to handle that.

(Figure 4)





(Figure 5)

Again we have to repeat the steps for the mobile version of the application. We have to re-write the test cases and DSL, UI object repository and execute our test cases to see that the application is working and compatible on various devices.

So we can see that there are two application forms available to us and two frameworks to do the test automation. We must remember that the two application forms are of the same products so the features, UI, business actions will be much similar. Still we have to use two different framework to do the test automation.

Also we need engineers who are not only good in their domain and testing but they should also have basic knowledge about test automation and programming. They should be able to write test cases in that format that the test automation tool should be able to understand it. Training them takes time and we do not have such time luxury in an actual project environment.

Solution is SMART

At QA InfoTech, through our extensive research and development we have created SMART which is

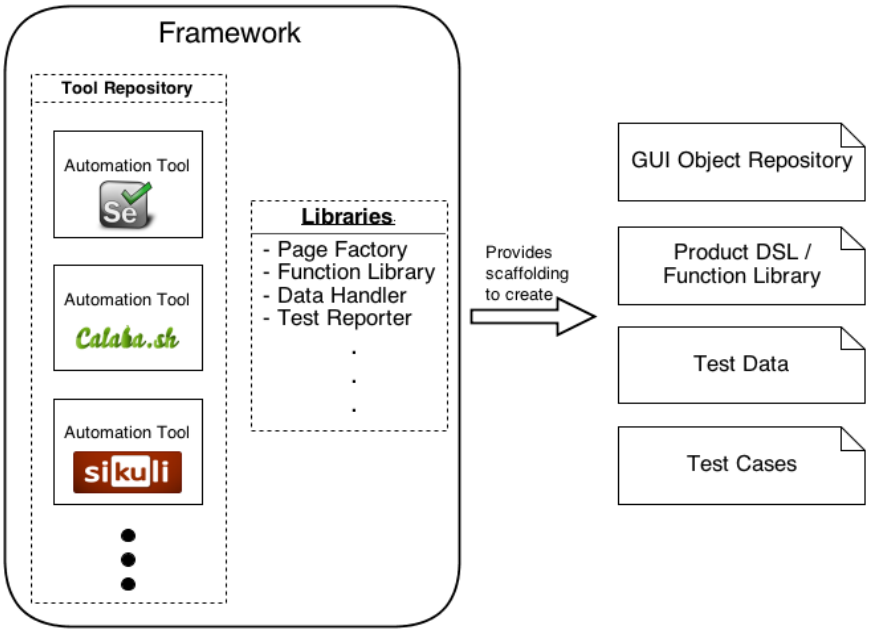
-unified automation framework which is amalgamation of various technologies

-capable of automating a web application and native apps of android and iOS

- is based on Behavior Driven Development in which we can write our test cases in plain English language.

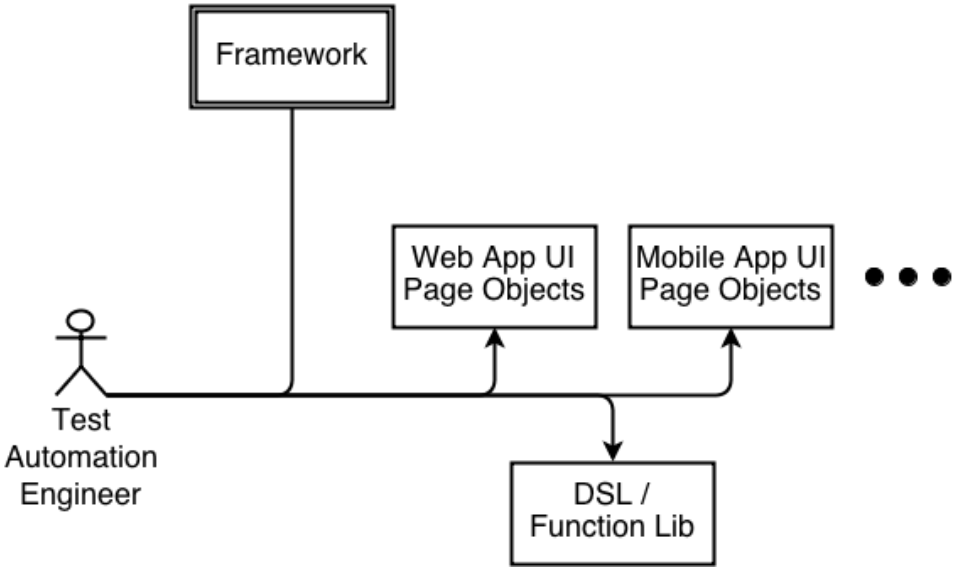
-is based on open source technologies.

Structure of SMART framework

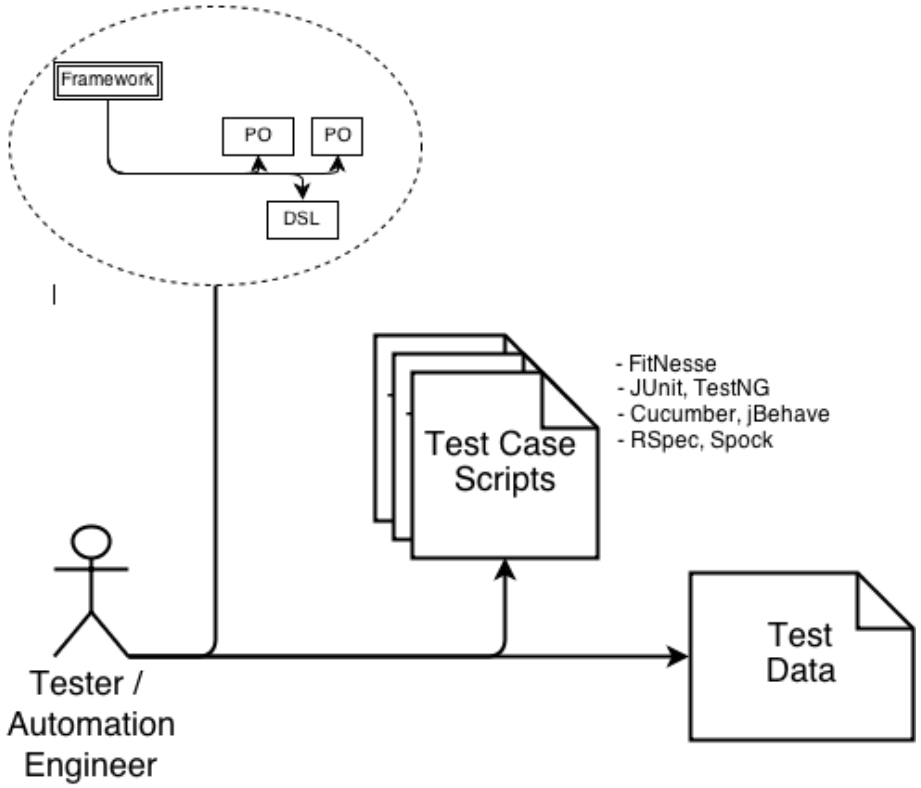


(Figure 6)

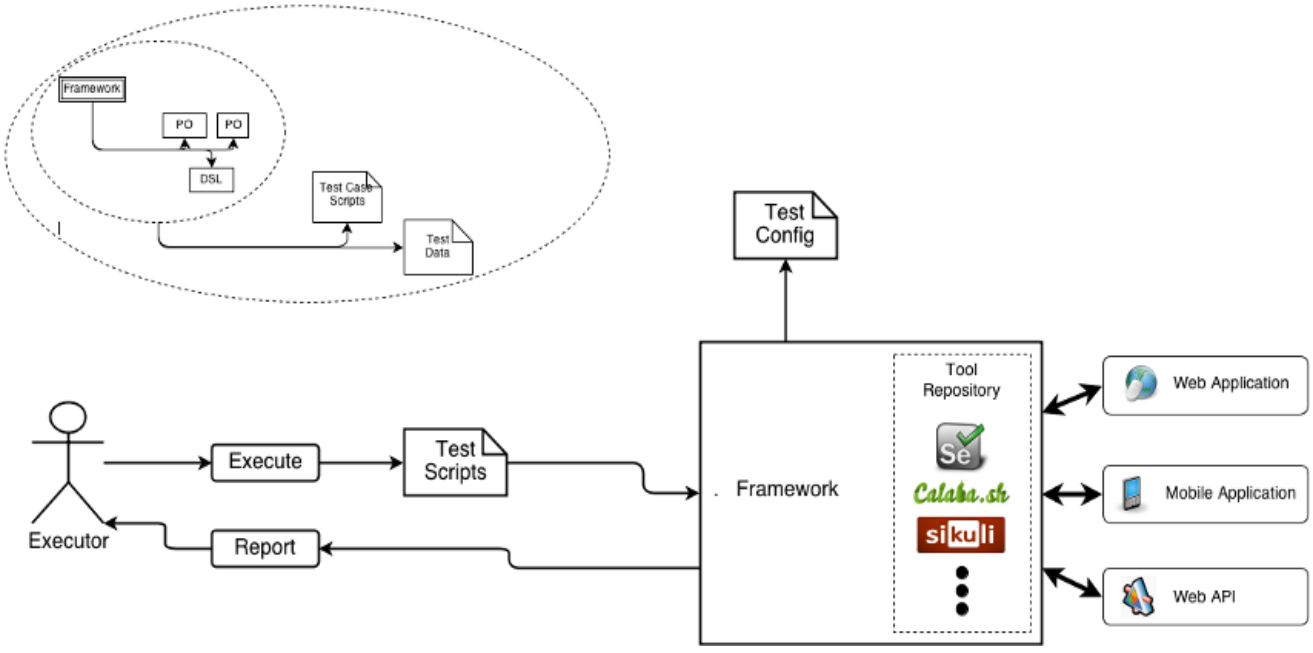
Step by step approach of using SMART framework



(Figure 7)



(Figure 8)



(Figure 9)

Writing your test cases

@Test001_TestVideoCreationFunctionaltyOfAnimoto

Feature: It should be possible to create video with selected theme

Scenario: Create Video

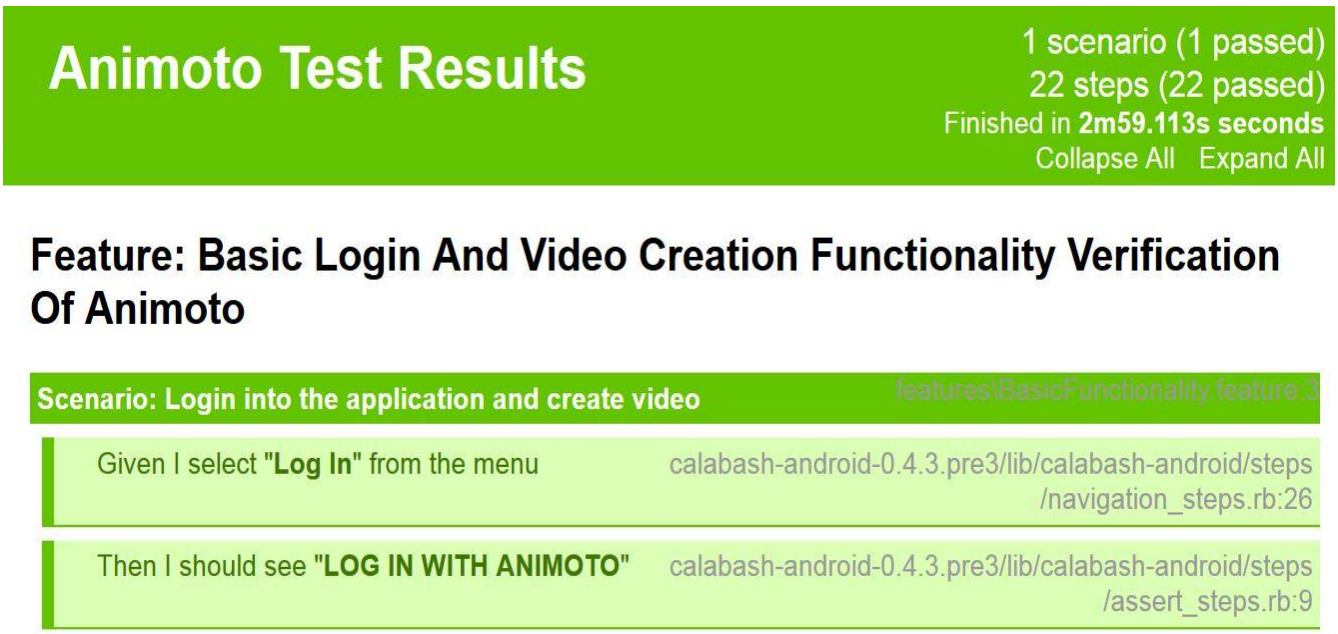
Given I navigate to animoto homepage

When I login as valid user

And I add pictures to create video from picture collection

Then I should be able to create video with selected theme

SMART Reporting



(Figure 10)

Tools Used in SMART

We have used the best tools that are not only used in QA InfoTech, but the whole testing industry is using them.

Web Automation Tools

Selenium Webdriver – Selenium is a free and open source tool. It is primarily used for testing and is also cross browser compatible.

Cucumber – cucumber supports behavior driven development. Users can write test cases in plain English Given, When and Then form

Maven – Maven is a build and dependency management tool.

Junit – Junit is used for writing unit tests

Java – Java is a free programming language.

Sikuli - Sikuli is an image based automation tool in which actions and verifications are based on images

Mobile Application Tools

Calabash – Calabash is an upcoming tool which can support both iOS and Android native applications.

Robotium – Robotium supports native Android applications.

Sikuli - Sikuli is an image based automation tool in which actions and verifications are based on images

SMART Benefits

- Test workflows span across multiple application forms.
- Parallel testing possible on both web and mobile platform.
- Continuous integration capable.
- Expressive business readable test cases.
- Cloud computing ready.
- Extendable, flexible and is based on open source technologies!!

Back To Index

Reference Material

http://en.wikipedia.org/wiki/Test_automation

http://seleniumhq.org/documentation/

http://sikuli.org/docx

http://fitnesse.org

Anmol Bagga is working as a Software Testing Engineer(Level 2) – Automation in QA InfoTech. He is an active speaker and has participated in various workshops, competitions and conferences. He has also trained and mentored many budding testers. He is strong believer of free and open source software.

Anmol holds a Master of Science degree in Informatics from Delhi University and Bachelors of

Science in Electronics (Hons) from Delhi University.

He can be reached at anmolbagga@qainfotech.net

Sharing is caring! Don't be selfish ☺

Share this issue with your friends and colleagues

Rants & Ramblings of a Mobile Tester

- by JeanAnn Harrison

Developing a Test Strategy for Mobile Software Projects

Planning out a mobile project can be a daunting task usually because little is known. Developing a test strategy is even more daunting considering few requirements are written for testers. Most requirements for mobile software are not documented, design is done in a "fly by the seat of our pants" style. Simply put, mobile testing is more complex than most desktop and web applications as interdependencies directly influence software behavior.

One problem not acknowledged or addressed in mobile projects is, what kind of mobile software application is being planned? Many who start the planning do not understand the difference between mobile native applications versus mobile hybrid applications. Many software testers do not understand the differences. So let's define what kind of mobile projects exist.

They are:

1) Mobile Native Application
2) Mobile Hybrid Application

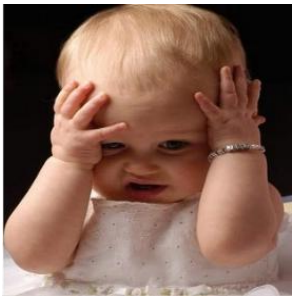
3) Mobile Web Application

4) Mobile Website

A big misunderstanding is to apply the same kind of approach to all four different types. This problem includes those who are testing, those who are developing, and those who are managing the entire project. But is it a wise choice to apply the same kind of testing to all four? Does it make sense or worse, efficient to apply load testing to a native or hybrid application? Does it make sense to apply hardware conditions like battery charging, clock/timing testing to a mobile web application or mobile website? Sure, you want to test a mobile website to make sure you can view a mobile website, but do you care about the clock values of the device to have an effect on your data displayed? Maybe. Depends on what the application is designed to do.

I often speak about developing your testing strategy before entering into a mobile testing project. What do I mean by "developing your test strategy"? Do you know what kinds of tests should be applied in your project? Not only do you want to consider what types of tests are appropriate, but you should also consider sequence of those tests. Do you apply automation? If so, to what? Why? When? Where? Not all testing within a mobile testing project should be automated, that is if you're goal is sufficient test coverage for your customers experience to be satisfactory. Remember, mobile users only take seconds to make decisions about their mobile software. If your brand is dependent on those who use your mobile application, then it would be a wise business decision to include your tests surrounding user experience. Otherwise, the business and brand can be lost in a matter of days not years or even months. Testers & test managers must think about the impact on the business while considering test strategies.

Where do we start to design a mobile strategy then for mobile projects? Here's a list to inspire you to develop your own list. This list is not meant to be the one and only application in developing a mobile test strategy. Each situation is different. Consider your company culture, your company's impact on the market, your competition, your project timelines, resources available to you, and in particular at the time of the project scheduled.



1) Define your mobile software project. What type of mobile software are you going to test? Is it a mobile native app, hybrid app, mobile web application or a mobile website? Is your project a combination of these mobile software types?

2) Have stakeholder buy-in on definition of mobile software project. Do your stakeholders also understand what type of mobile software will be developed and tested?

3) Define your test case types. Create a list of the types of tests on a spreadsheet (or your favorite organization tool) which could have a direct impact on user experience. Note: test types refers to functional, behavioral, security, data validation, performance, usability, trainability, GUI look & feel, etc. What parts/areas/functions of the application would you want to apply these test types? Identifying where to apply test types can considerably lessen your testing burden. Why not consider combining test types so you can cover more testing in less test steps?

4) Define where you can apply automation as you cannot apply automation until you have defined what you want to test. Examine what tests you want to apply keeping in mind scope, risks, and efficiency. Do not make a decision to do automation for the sake of automating. Use automation to allow you more time to perform exploratory testing and user experience testing. These kinds of

tests take longer to perform and are manual for the most part, yet can have a massive impact to uncover bugs otherwise would not be found by automated scripts.

5) Determine your Exploratory Testing sessions throughout the project. When little is known about needed requirements based on how a software application can work with the device's firmware and hardware, testers can provide a great deal of information to stakeholders. But the time to conduct your Exploratory Testing can save a company's reputation, maintain a market share against the competition, or simply save the company a lot of money. Knowing more about how an application will behave prior to release is vital to today's instantaneous mobile market.

6) Determine which tools you will require you to implement your tests. Consider acquisition and set up of the tool environment which may require obtaining new hardware, requesting outside your department resources and training on the use of the tool itself.

7) In your spreadsheet, include risks based on the test types chosen, the scope of the project, available trained resources and project schedule. Consider the risk for the decision to not test. What effects of this decision will have on company reputation in the market, competitors, application recovery, etc.

8) Also on your spreadsheet, assign priority & severity based on stakeholder's needs and expectations. This task will help in the decision making of what to test and what types of tests to conduct within the scope of the project.

9) Include necessary training time of resources as needed. Training can include domain knowledge of the application, device training, test design, how to implement the various test types, execute the tests (manual and automated) as well as analyzing the results.

10) Have a stakeholder review of your spreadsheet. Review your test types, tests, choice of implementation of the tests, risks, prioritization with all the stakeholders including development, marketing, sales, and customer service. Ask questions, define risks, engage your stakeholders, and challenge them to accept the risks.

What kind of strategy then could work for each mobile software project?Remember, this article is not designed to provide a step by step bible-like instruction set and instead a general guideline to inspire what will work best in your domain. Test strategy for a mobile website project is different than a mobile native app project. After all, would you define load the same way for each considering the CPU processing is physically different? What about stress testing? There might some overlap but it's not a copycat process by any means.

First of course, let's distinguish between "mobile web app" with a "mobile website" as many confuse these are the same. But I define a mobile website as displaying information and does not depend on user interaction. Mobile websites or desktop websites can interact with the user like filling out a form but the user isn't required to fill out a form or enter information in order for the website to fulfill its purpose. A mobile web application or web application does depend on user interaction to fulfill its purpose. User experience, therefore, is different between these two types of mobile projects or software projects. An easy way to understand the difference, look at your favorite restaurant's mobile website and compare with a retail website like Amazon.com. Would you apply the same tests to both types of mobile entities?

Mobile Website Test Strategy: Test types applied to mobile website project would center on visual experience. Things like font size, colors, placement of images and text will be high priorities for the stakeholders. Testing various browsers used on devices will become highly important with overall performance. Does the website load quickly? What kind of bandwidth is required? Software test types center more on the display of the website itself from design, compatibility, accessibility, and usability. Less focus is put on tests regarding security, behavior of the functions, and dependence of the device CPU to do any processing. There is little need for exploratory testing so more automation can be applied. However, keep in mind every situation is unique and perhaps one test type is higher priority for one website than it is for another website. Depending on the expected traffic to the website, you might want to make sure you can reach your baseline of expected traffic at any given time. Automation tools can help so you will need to assess which tool(s), who is trained on the use of the tool and what kind of resources you will need for the testing. Assign priorities to your tests, get buy in from stakeholders and implement.

Mobile Web Application Testing Strategy: Test types here will require more focus on functional testing, data validation testing, as well as overall performance. Obtaining benchmarks of access, network communication and load on the network server's CPU would be high priority to your test effort. Here again you will want to consider a tool or perhaps more than one tool to get the information about the web app behavior. Once you introduce a tool or tools, more considerations like training, set up accessibility to the tools must play a part in your strategy. Mobile Web Apps are complicated to test since Performance Testing is a huge part of the User Experience. Attention to security, data validation, load, stress (on several levels), functional, network communication, browser compatibility, carrier tests will have strong focus. AddUsability, trainability, desirability, testing the value and usefulness of the web application to the testing effort which creates more complexity than testing a mobile website.

Let's define both the Hybrid App and the Native App. I like Julian Harty's definition: "a Hybrid app is a combination of native code (often written by a person, but may be written by a code-generator or provided as part of an app generated by a code-generator) and a webview control embedded into that app. So the app includes two technologies - a webview (where HTML, JavaScript, CSS, etc are rendered and processed) and native code which provides the rest of the functionality and UI."

A Native App then is purely contained within the device. All code is downloaded to the device, all processing is done from the device's CPU and the app does not contain HTML, JavaScript or CSS code. The GUI displayed does not utilize a web browser or connection to the internet. A native app can interact with the internet but only for the purpose of sending data from the device to another application outside of the device. The native app is a self-contained application dependent on the device's firmware and hardware to function.

Based on these definitions, one can imagine the different approaches to developing tests. Architecturally, the hybrid app is very different from the native app with completely different inter-dependencies. This is often the failure in mobile test projects. Would you apply the same tests to the Expedia's application as the alarm clock on your device? Does the clock app send a message to ask for the time on some web server? No, it utilizes the internal clock, hardware inserted into the device. So performance testing the clock would not involve testing how many users can access that app at any given time.

Mobile Hybrid Apps: Developing a list of test types for testing the hybrid app will greatly depend on what the application does, when and where the application utilizes webservers. However, hybrid apps also interact with the device's hardware and firmware. The mobile tester therefore must understand the architecture of the application in knowing where and when the code interacts with native code or uses webview code. Basically a combination of mobile web application and mobile native app test strategy will suffice in testing a hybrid app. The more the tester knows about the architecture, the more thorough tests will be, a higher level of test coverage can be achieved. Because of the complexity of the

architecture, testing the hybrid app immediately becomes intricate. Webserver and native device conditions add to the difficulty of test design. Testers will be required to do much more Exploratory Testing in order to learn how various conditions affecting software behavior. The application's behavior has not been determined because much is unknown. Requirements are not as defined prior to the first release but this can be a deadly strategy for an y company trying to make its mark in their market. Speeding up the CPU through data searches or displaying large amounts of graphics, communicating with webservers, dependencies with carriers, charging the device, and more all can have a direct effect on the Hybrid application's behavior.

So the tester will need to build in enough time for exploring the hybrid behavior which is why knowing as much about the architecture as possible will help facilitate the testing. Prioritizing the testing will also be critical considering the short timelines as most stakeholders do not realize the complexity of hybrid app testing, so it's critical for testers to take the time to work out the test types and work out the risks with the stakeholders to get their buy in on what will be covered in the testing per project.

Mobile Native Apps: Developing a test strategy for native apps is very different from desktop applications or web applications. Testing native apps is often misunderstood by testers and stakeholders with regards to planning out the project. Testers will have to functionally test the software, consider usability, how easy to use the application is vitally important to the user experience but also how hardware and firmware conditions directly affect the behavior of the app itself.Performance testing is defined very differently for native apps as compared with hybrid, mobile web apps and mobile websites because only one person is accessing the application directly on the device. But how a native application performs while the device is charging is often overlooked by testers. I've seen tests reveal bugs including data corruption, timing, memory leaks, poor or nonexistent network connections, rebooting of the device. Native apps do not have the same dependency when it comes to security because interaction outside the device is not part of the functionality. Therefore, native apps might send/receive data via wirelessly but a high priority on security tests is rare. Software testers need to think differently when it comes to "performance testing" on native apps and plan accordingly. Native apps will require a lot more time in Exploratory Testing because so little is known in how an application would behave with the device hardware and firmware. If were talking phones and tablets, there is new models of hardware and new versions of firmware which will require new tests. The biggest problem with native apps, assumptions are drawn where little testing time is needed because the application is contained. Stakeholders have less understanding for the need of longer timeline for the project.Many test managers and testers believe automating the GUI display and functionality tests are adequate test coverage of the native app. This assumption is the biggest risk. Remember, native apps exist on an entire system, and the entire system needs to be included in the test design.

Taking the time to examine what kind of mobile project, develop architectural knowledge and inter-dependencies is necessary by mobile testers. One major factor in test strategies is to also consider the sequence of functionality and how that functionality works with various conditions of the hardware and firmware. For example, does the mobile application always communicate wirelessly while charging? Under what variations can the mobile app not communicate wirelessly? Requirements rarely cover these kinds of questions which is why developing a mobile test strategy is necessary. Mobile testers need assistance from their stakeholders, whether it's to get acceptance of what testers can cover based on the project time line or for stakeholders to re-examine the project timeline and release date. Project stakeholders including testers all want and expect to be successful in releasing the software.

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

CONTINUED FROM PAGE 13...

How important it is for a Test Manager to have hands on experience? How do you find time for it despite of having so many other things to deliver already?

Many people want to challenge the decisions a testing team makes in order to reduce the time dedicated to testing activities of which they often do not understand. As such, I do believe it is important for a Test Manager to have hands-on-experience because you will need to defend why it is not prudent to reduce testing. When you have hands-on experience you understand the risks of eliminating testing time and how that can impact the quality of the release. As a personal rule, I have never asked my team to perform any type of activities that I would not take on including extra hours when needed. When timelines are tight, it is important that Test Managers do not hand out assignments expecting the team to work late but then leaves for the day. We must be a strong team! I do have a lot of responsibilities that has recently increased which, results in less time for hands-on-testing. But there is always time to discuss strategy, test approaches, training, career development, and any other issues the Testers may wish to discuss.

Three things that you do without fail (as a Tester) on daily basis...

It is important to review the progression of assignments to understand any risks, challenges, and ensure you understand the top priorities since they often change.

A daily review of the workload versus deadlines is important to ensure you do not find yourself without sufficient time for your assignments. Lastly, I review the short-term deliverables, future deliverables, and strategic direction of the company. It is best to start preparing your team now for change and to identify the knowledge and skills that the Testers might need to address sooner rather than later. It is about balancing time because you cannot be focused solely on today - it is important to be prepared and ready for the future.

Your other blog 'Realistic Cooking Ideas for Busy People' has become quite famous and there are many awesome recipes that you have written.

Please tell us more about your love for cooking.

Thank you for asking about my cooking blog. I enjoy cooking as a hobby and it is a creative outlet that includes exploring cuisines from around the world. Last winter my focus was Indian cooking and all the wonderful spices that adds tremendous flavor to the dish. Another winter had a focus on Asian cooking with using a wok to explore new recipes. My cooking is based upon what my husband and I enjoy and my recipes will not be authentic to a particular nationality but their ingredients inspire me.

As an example my Indian Chicken Curry soup may be different than what you will be served in an Indian Restaurant. But it is one of my favorite Indian dishes. I first had it on a hiking trip in New Hampshire and immediately fell in love! Each night I ordered the soup trying to figure out the ingredients and asking the server what was in the soup - then taking careful notes. Through research and reviewing my notes, I came up with a recipe.

By nature I am a learner and through cooking and testing. There are always learning opportunities. Plus I am not afraid of a challenge and figuring out a recipe for Indian Chicken Curry soup was a fun adventure! Whenever we go on vacation, I like to try different dishes in order to make a wish list of recipes. It is great inspiration and is a lot of fun!

Per my personal experience testing slowly develops low tolerance towards buggy things while cooking requires more practice and patience. How do you balance your both of these interests, which appear to be opposite in nature?

I actually consider them very similar - my testing approach is exploratory in nature as is much of my cooking. I research recipes and ingredients to come up with my version. When making the recipe, I will perform taste tests, which may influence adding more or less of an ingredient. Each time I make a recipe it might change based upon ingredients that are in the pantry and through my taste tests. Similar to testing approaches and techniques, I research different ingredients and ways to cook (such as slow cooker, braising pan, cast-iron skillet). Then when an opportunity arises, I tap into what I learned to apply it - similar to testing.

How important do you think are certifications for Testers?

Personally, I believe that Testers, and really anyone, needs to be careful of what type of "initials" you put next to your name. If the "initials" do not have credibility it reflects poorly on you. If you pay $49 for a certificate where you can look up the answers to pass the test, I would wonder about your dedication to the professional testing field and how you would handle the complexities of testing.

From the medical field I like how physicians are trained with the approach: See one, Do one, Teach one. I use a variation of this approach. As an example, someone demonstrates how to test an area of the product and then you perform the testing on your own. Once you believe you understand the material, you either teach another Tester or walk through your testing steps with the trainer.

I like this approach to learning because it is the application of what you learned with the ability to either teach or demonstrate your understanding of the material.

When you interview a candidate for a tester's profile, what all things/skills you look out for?

The most important thing I look for is fit to the department and the culture of our organization. The candidate could have excellent testing skills but if he will not fit within our environment then I will not hire him. We have a strong team environment and a bad fit can be a real problem. If the candidate is a fit for how we work, then I look for someone who is teachable from testing and technical skills.

You have been recently promoted to a new role in your organization. Would you like to tell us more about it?

My career has been focused around data quality. I have been employed in different industries; however the need for quality never changes regardless if it is banking, clinical data, or supply chain data. The customer is making decisions based upon the reports and information provided from your solution.

I was hired as the Software Testing Manager and earlier this year I was promoted to Director of Quality Management Programs. My new position provides an opportunity to be involved in company-wide quality initiatives while managing the Software Testing department from a strategic perspective. My personal viewpoint is that quality drives an organization from a holistic approach with everyone having a role in providing a quality product and excellent customer service. This opportunity provides an avenue for me to have a larger influence on company-wide change and being part of strategic solutions.

You have been writing several articles on software testing via different testing magazines and forums. What is your inspiration for writing?

When working through a problem, I find that writing my thoughts and laying out visual pictures such as mind maps helps challenge my own approaches to find better ways to address a problem. My inspiration for writing is partially because it challenges and sharpens my testing skills, knowledge, and approaches. Plus I enjoy sharing my knowledge and experiences with other people and there is a lot I can share from a technical and business perspective. I learn a lot from other Testers and it is important to me personally to contribute.

What would be those books, blogs and related forums you would want your team to regularly read/follow?

It is important to follow the well-known leaders in this field that includes but is not limited to: Jerry Weinberg, James Bach, Michael Bolton, Lisa Crispin, Cem Kaner, and James Whittaker.

But also follow the testers who are in the trenches. They may not be consultants, own their own business, or travel the conference circuit to present. Many of them are leaders in their own right and are influencing the Testing community.

Recently on my blog "The Testers Edge" I wrote about "What is your Legacy?" We each will leave a legacy - what will your legacy look like?

I ask all new Testers to subscribe to Tea-time With Testers, as it is important to understand how other Testers are approaching similar problems.

Your message to testers who want to bring change in their testing culture would be....

Understand your circle of influence, which basically means what change you can influence. Do not be concerned if you cannot make change outside of your department. Too many people get discouraged because the problem is larger than what they can change - often because they need to change a process that crosses departments. Instead look for a small change that you can make and build upon that success.

You have been writing for TTwT for over a year now and we can't thank you enough for that. How do you feel about this association? We would like to know your feedback and suggestions.

I thank Tea-Time with Testers for the opportunity to write for your wonderful e-magazine. I always admire it and am honor to be a part of the TWT family.

My suggestions are to keep the magazine fresh with new writers and ideas on how to approach the complexity of testing. Never become predictable. Keep them guessing on what is coming next!

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

The shattered tube light

My younger son was playing with bat and ball in his room. A young kid with a lot of energy, he hit the ball hard, it rebounded off the wall, targeted the tube light, smashing it into smithereens. Now the onerous job of cleaning of room fell on me. I swept the floor first, picking up the big shards, and then vacuumed the floor and bed multiple times to ensure that the tiniest of the fragments have not been left behind.

Being a tester, the question that I had was "How do I know if there is no broken glass left behind?" When I swept the floor, I could see the broken ones that I collected and disposed, could hear the bits of glass being sucked up when I vacuumed the room. So does it mean that when I do not see or hear, the room is clean? Similar to the fact that, when I do not find defects in the software, it is assumed to be clean.

Nah! That I do not see or hear, implies no glass shards doesn't sound convincing. So I systematically cleaned the room covering all the areas, all the visible surface area, under the cot, study table, the bed and then thought like an "user" i.e. 'what would I walk/touch' and covered these areas too. As I constantly refined to 'cover more area'. Once in a way I would hear the sound of glass being sucked up by the vacuum cleaner and that was reassuring.

So where am I going with this story? That being convinced how well we have cleaned the room is a better judge of cleanliness and safety, rather than feeling good when we find glass shards. That just information about defects ("glass shards") is not good enough for judging quality. That it is very necessary to know how well we have swept the room to be confident about cleanliness and safety.

Most often defect information is used to make judgment of quality. We use the rate of arrival this. If the number of defects found decrease with time, we infer that the quality is improving. Let us deliberate upon this. Defects become visible when failures are noticed, which are the result of irritating latent fault(s). And this happens when we 'sweep' the system with a set of inputs i.e. test cases. And if we have not covered the important areas of the software equivalent to 'sweeping the room', then we may not surface these defects. And therefore no glass shards, until it pokes us - Ouch.

So it is imperative that we are confident about how well we have swept i.e. have we covered the areas that matter completely? Are we looking for all kinds of shards? I.e. different types of defects. A rational analysis of the area covered and test coverage, and being able to justify the completeness of test cases is very necessary to really judge the quality of the system.

The confidence of good quality is really about the intrinsic measure of goodness of test cases rather than the extrinsic measure of defect count. So what is the defect information useful for? It is great to enable continuous refinement of the test cases to improve coverage and to prioritize work. That is based on the severity and priority of defect(s), we can make a logical decision as to where to expend effort, and wisely use the time/effort. So how can we make figure out we have 'swept the area' well? What is 'area' in the context of software? Area in software represents the entities being tested - features, requirements, flows or at the lowest level code fragments/modules. Once we are clear and able to setup a clear baseline as to what is to be tested, we move to figuring out 'how well have we swept or covered'. And this means we need to logically justify the type and number of test cases. What types of test are we performing, and how do we justify the number of test cases for each type of test for each entity. I.e. what kind of glass shards are we looking for and how do we know that we covered the entire area where we are looking for?

So how do we justify the number of test cases? By understanding the various conditions that govern a behaviour, that when combined optimally indicate the number of test scenarios that we should look for. And then an optimal combination of number of test inputs for each scenario enables us to assess the completeness of test cases. I.e. how well have we swept. And do remember the number of test cases is not static, it does changes with time as we understand the system better and morphs to be in sync with the evolving software. It takes a good set of test cases to find defects that matter. So measure the goodness of test case to judge the quality of final system. Start by logical justifying as to what you are looking for, in-what and why that many number of test cases are will indeed cover the area in which you are looking for.

"It is not faith in the outcomes, but strength in the doing" that enables clear judgment of quality.

On an end note, I sat down with my son and explained once again as why he should not play wild shots indoors. He nodded quietly and promised me he will be play softly. Deep down, I have a nagging feeling that this is not the end. And when it happens again, I hope I can learn something that I can apply to testing software!

So if you can be sensitive and not create some of the defects in the first place, the room does not get dirty. But…

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**

testing intelligence

- its all about becoming an intelligent tester

an exclusive series by **Joel Montvelisky**

Are we all Accidental Testers?

Two weeks ago I was giving a presentation at StarEast. I will write more about the content of this presentation on a later post, but I wanted to write about something that happened "on the side" of it that made me realize an interesting characteristic of most testers out there in the world.

During the introduction slides I mentioned that I was a case of being an "accidental tester", someone who had gotten into testing more by chance than as part of a conscious and planned decision.

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?

This was not an integral part of the presentation, but more of an introduction to get the audience to understand from what point of view I wanted to present my ideas.

Then, after the session was done, two different people came to share with me that they also saw themselves as accidental testers and to tell me how they had personally ended up as testers without having intended to do this in the first place.

Does anyone ever plan to become a tester?

After talking to these two "accidental testers" I kept asking people in the conference about their own personal paths into testing, and I found that everyone I talked to had stumble into testing from different sides of the working spectrum.

There were those who had started in the business side and moved into testing as a side-effect of having displayed technical interest and abilities. There were those who had begun as tech-writers or support engineers, and moved into testing because they saw it as a more appealing job. I also talked to some who had come from a programming background and had been asked to test in a specific project, only to understand that they enjoyed this far more than their programming tasks.

In short, I don't think I found even a single person (although I am sure there were) that had made a conscious decision to become a tester as part of a planned and defined career path and profession.

What should we do about this?

Should the fact that most of us ended up as testers by chance bother us?

I think not, and not only that, I think that people who end up in testing as part of an evolutionary path are more prone to be better at their jobs.

Why is this? Because they progressed into their jobs based on their natural strengths and not due to a decision that sometimes may be artificial and not in-line with their internal (intrinsic?) personal capacities.

For example the people who went to law school or to study accounting in order to please the expectations of their parents, or those who went to study management because they could not choose any other profession from the list provided by a university, or those who studied computer science because they thought it would provide them a high salary.

We are good at what we do!

We are testers because we like to test, and many of us are even pretty good at it  I don't know many testers who endure working for long in our profession otherwise. We should not let the fact that we did not get into our profession following a "standard path" affect our self esteem in any way. A good tester needs to be a self-learner by nature, he won't have a chance to survive otherwise.

We need to constantly learn our AUTs and how to test them effectively. We need to learn how our users approach our applications and how they use them as part of their daily activities. We need to learn how to effectively communicate with our stakeholders and peers in order to gather information and then to pass it along to all the team. And we need to learn many additional things that are simply not taught in any formal way in any university or learning institute.

Self learning is another one of our testing tools, and maybe the fact that we are "accidental testers" is actually part of the natural selection process that helps define who will be a good tester, and who may do better by looking for another profession for him or herself.

So, the next time you feel like you ended up in testing by mistake and that maybe it was the result of a wrong turn along the way, look into this thought again.

The fact may be that you were a tester from the very beginning, but you needed to go through a self-development process in order to reach your professional maturity and fulfill your calling as an exceptional 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

Advertise with us

Connect with the audience that MATTER!

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

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

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

Well, hear directly from them.

And the **Good News** is…

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

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


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

in next issue

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 - A Thoughtful Eye

Core Team:

Dr.Meeta Prakash (Bangalore, India)

Unmesh Gundecha (Pune,India)


Dr. Meeta Prakash


Unmesh Gundecha

Online Collaboration:

Shweta Daiv (Pune, India)


Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)


Kiran Kumar


Chris


Romil

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

To get **FREE** copy ,

Subscribe to our group at

Google™

Join our community on

facebook

Follow us on

www.teatimewithtesters.com