

Testing Circus

Volume 4 - Edition 9 - September 2013

*3rd Anniversary
Special Edition*

Interview with
Jay Philips

**Your Monthly Magazine
on
Software Testing**



- A Fake Tester's Diary
- How to Find A Good Tester
- 40 Definitions of Testing
- Autobiography of Automation
- User Experience Testing
- Is That Testing?

www.TestingCircus.com

Thanks to all those testing services companies who claimed to save millions of dollars for your customers by automating manual tests. Some of your customers are coming to us with the money you "saved" for them.



brainual software testing services

sales@moolya.com

**3rd Anniversary
Special Edition**

Volume 4 - Edition 9 - September 2013

Testing Circus

Topic	Author	Page #
Letter to the editor		5
Editorial	Ajoy Kumar Singha	6
How to Find A Good Tester?	Trish Khoo	8
Learning To Test With Risk	Mike Talks	11
Are You Listening to Your Intuition?	Bernice Niel Ruhland	15
Autobiography of Automation	Yagnesh Shah	17
Heuristics – User Experience Testing	Dheeraj Karanam	22
Security Testing for Beginners	Nagasahas Dasa	23
Research ReCycle Reseed	Trinadh & Indrani	27
Is That Testing?	Katrina Edgar	37
Tips and Tricks for Testing Cloud Deployment Software	Vivek Sharma	39
41 Definitions of Software Testing	Jyothi Rangaiah	48
The “CMB RSM TCP” Software Tester	Ajay Balamurugadas	50
Impact of Requirement Changes on Business and Quality?	Rrajesh Barde	54
Is It Time to Do Away with Low Level Test Scripts?	Praveen Bhandari	61
Books Recommended by Testers	WoBo	64
A Fake Tester’s Diary, Part - 32	Fake Software Tester	66

Topic	Author	Page #
Testers to Follow	--	70
Crack The Code	Blindu Eusebiu	72
Interview with Jay Philips	Ajoy Kumar Singha	73
Security Testing Tips (Part 9)	Blindu Eusebiu	77
Learn Sahi Step by Step, Part 7	Narayan Raman	79
QTP Code Corner	Naina Dhiman	86
Optimize System Integration Testing Efforts	ToolsJournal.com	91
Solving Data Driven Test Automation Mysteries	ToolsJournal.com	92
Towards Test Automation, What & What Not To Automate	ToolsJournal.com	94
Testing Events Around the World	TestEvents.com	96
First Look at Our New Website	Team Testing Circus	99

*On the Cover Page - Jay Philips

Testing Circus Team

Founder & Editor – Ajoy Kumar Singha

Team -

- Jaijeet Pandey
- Srinivas Kadiyala
- Pankaj Sharma
- Naresh Bisht
- Bharati Singha
- Chanderkant Saini
- Dhakshna Moorty P

Editorial Enquiries: editor@testingcircus.com

Ads and Promotions: ads@testingcircus.com

Other enquiries: info@testingcircus.com

Testing Circus India

Chaturbhuj Niwas, 1st Floor,
Sector 17C, Shukrali,
Gurgaon - 122001
India.

Volume 4 - Edition 9 – September 2013

Testing Circus. Published from Gurgaon/India.

© Copyright 2010-2013

ALL RIGHTS RESERVED. Any unauthorized reprint or use of articles from this magazine is prohibited. No part of this magazine may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system without express written permission from the author / publisher.

Edition Number : 36 (since September 2010)

Three years ago, Testing Circus was a newbie testing magazine with a handful of testers contributing articles and a few hundreds of testers reading them. Today, it is one of the most highly respected magazine in the world. In last three years, Testing Circus has published articles from testers who were first-time writers, testers who had something new to share and testers who published their failure lessons so others could learn from it. In short, Testing Circus has become a platform for testers who dare to tell their story fearlessly. I have personally liked the Editorials, A Fake Tester's Diary, Testers to Follow and Interviews. It gives me immense joy and pleasure to see Testing Circus complete three years in this great journey of educating testers and others alike. My heartfelt thanks to entire unit of Testing Circus for the great work done and for the future ahead. Congratulations! Rock On!



~ Parimala Hariprasad
<http://curioustester.blogspot.com>

I'm always happy to see a new issue of Testing Circus, though it often takes me awhile to read an edition, with so many competing priorities. I enjoy reading the latest ideas of old friends, such as Bernice Niel Ruhland, Michael Talks, and Rob van Steenberg just to name a few, and also learning from people I didn't know before, such as Jayshree Rathod. I enjoy the recurring columns such as " A Fake Software Tester's Diary", and the interviews such as this month's with Iain McCowatt. We're so fortunate to have more resources these days to learn more about testing, and more about inspiring testers around the world. Congratulations on three years!



~ Lisa Crispin
<http://lisacrispin.com/>



From the Keyboard of Editor-in-Chief

I am happy to present the 36th edition of Testing Circus which marks the completion of awesome 3 years of this magazine. To celebrate our 3 years of publication, we had to do a special edition and here is a 100 page magazine with lots of testing articles both from experts and first time article writers. We have also released a free ebook on how to promote software testing in an organization by Rob van Steenbergen.

This also marks the launching of our new community based website. With the new website you will be able to participate in discussions, connect with other testers and join special interest groups. We hope you will join our community and participate in meaningful discussions. I would request all of you to register and be a part of the community.

On this occasion, we would like to thank our supporters Moolya, TestMile, Tyto Software (Sahi) and Talent Plus Plus for their continued support. We would also like to thank all our article contributors without whose support continuous publication of this magazine would not have been possible.

Enjoy this wonderful edition of Testing Circus. We welcome your feedbacks to make Testing Circus more readable and testing community friendly. Write to us at editor@testingcircus.com

Like all other applications, our new site is also not exhaustively tested. Should you encounter a bug in the site, consider to submit it to us. There a few more enhancements that we have planned to release gradually in coming weeks and months.

Keep reading Testing Circus. More exciting news next month. Happy Testing!

- Ajoy Kumar Singha



Write to us at editor@testingcircus.com

WEEKEND TESTING

Looking for a cost effective
way to learn *hands on*
testing?

No, it is not free.
It costs your time on weekends.

OVER 100 SUCCESSFUL WEEKEND TESTING SESSIONS



weekend

Testing

test learn contribute

Visit www.weekendtesting.com for an
opportunity to life time learning.

How to Find A Good Tester?



- Trish Khoo

A few years ago, I was a Test Manager looking for a new tester at a company called Campaign Monitor. From previous hiring experience, I knew that it can be tough work to find a good tester. Writing job descriptions is something I've always found a bit difficult. Luckily, this time around I had a bit of help from **Anne-Marie Charrett**.

I didn't actually think it would be this hard, because it was actually a genuinely good job. Personally if I could have picked any testing job in Sydney it would certainly have been this one, hands-down. It's a good product, great people, a really beautiful office, heaps of benefits, flexible hours, a great salary and certainly enough challenging work to keep a good tester interested. And how many places in Sydney will offer you a private office and free lunch every day? So I thought this should be a pretty easy sell, but it turned out sales is a bit harder than I thought.

The job title

I initially set the job title to "Test Engineer", following the vague logic that "Engineer" was an attractive title to technical types, and perhaps it would attract some developer-testers to the role. I've also heard the job title "Software Engineer in Test" used quite often, which the cynic in me suspects is a way to get around the

misguided idea of technical testers not being "real" software engineers. But I can see the appeal in the title.

Anyway, after some thought I realised that the coding expertise of the candidate was not as important to my team as the ability to test and analyse the software. We have two automation specialists here in a team of three, and our automation stack doesn't have a very high maintenance cost, so if our third tester can help out that's great, but not essential. What we really need is somebody who can help with the exploratory testing of new features, and who is capable of shaping a risk-based test approach around the context of each feature in order to test it effectively and efficiently.

So "Test Analyst" is a much more suitable title. Given the amount of responsibility, control over the test approach and experience required for the role, "Senior" seemed like an appropriate prefix to that job title. In addition, one tester told me that the position was less appealing to her as an experienced tester, because it was not a senior role. So even though "Senior" seemed a bit redundant to me in a team of three testers, I guess it holds some weight.

The appeal

For some reason, it hadn't initially occurred to me that I would need to make this role appealing to testers

looking for a challenge. It took Anne-Marie to point out to me that my job description was boring for me to realise that I would probably not even apply for my own advertised role. I don't like generalising from self, but I thought that if I want to hire somebody like myself, I should write a job advertisement that would appeal to me, as a job seeker. So I thought, what would I be looking for in a testing role?

- An interesting product to work with
- A variety of different testing challenges, to keep me interested
- Influence over the testing process itself
- A pleasant work environment with good people
- A nice big automation stack to tinker with

So I revised the first paragraph to not be about what I want in a candidate, but what I could offer a good tester in terms of an interesting job. Then I went on to briefly explain the kind of candidate that would appeal to us. I hadn't even mentioned the private office, free lunches and other cool stuff, so I put that in too.

The bullet point list

Bullet points are great for this kind of thing, right? My initial bullet point list included such boringness as "Experience writing test cases and bug reports". I guess I was just trying to "widen the net" by making such a general statement, but it made it look like a very basic, unskilled role. Which is crazy, because actually we're quite selective and we want somebody with excellently awesome testing skills. So I expanded the bullet point list to show more specific points.

The extra homework

So now that I'd made this job look much more appealing, and decided to advertise on the employment feeding-frenzy that is SEEK, we needed a better way to whittle down the applicants. My company used a very innovative approach to hiring our latest designer. It's difficult to use the same approach for hiring testers, mainly because I don't think testers have quite the same online presence as designers do. But it was interesting to see that the best designers would put a bit of extra work into their applications for the role.

So we figured that if we ask for a little extra effort in applying, we could weed out those who weren't really interested and those who didn't read the description.

As well as this, it's a good opportunity to get an impression of the candidate's testing skill.

So, after all that, this is what I came up with. As it turned out it attracted a former colleague of mine, Adrian Lai, who took over from me as Test Manager at Campaign Monitor after I left and is still there today.

Senior Test Analyst

As a senior member of our test team, you'll be helping to shape our test approach, as well as testing new features and improvements for Campaign Monitor. At its simplest level, Campaign Monitor is a web application used for email marketing and analysis. However, dig a bit deeper and you will find that it is a complex system that offers a variety of different testing challenges. Some primary testing concerns are usability, security, performance, load, cross-browser compatibility, third-party integration and mobile devices, to name just a few. Testers are involved from design right through to implementation and release, making this a challenging role that will suit any tester with a special interest in improving software from day one of the development cycle.

Our ideal tester understands that there is always another question to ask; that nothing is exactly what it seems to be; that there are no absolutes in software. Here at Campaign Monitor, we are constantly evolving the way we work. Our testing processes and approaches are definitely not set in stone, we need someone who is able to reflect, adapt and improve with us.

We love people who are passionate about what they do and about improving how they do it. If you're motivated by continuous learning and improvement, then we can give you the tools and support that you need. We'll also give you your own private office, a machine built just the way you like it, free lunch every day and flexible work hours.

Ideally, we would like an experienced tester who can read and write C#, or a similar language. This is because we maintain automated test suites written in C#, and it would be great to have another hand on board to help us maintain and improve it. However, we will definitely consider exceptional candidates with little to no programming experience. But basic knowledge of HTML and CSS is a must.

Required skills:

- Ability to create and define a test approach for any given context
- Skilled at analyzing a product for testing, using a number of different oracles
- A good understanding of metrics and their use in project analysis
- Adept at writing robust test cases that stand the test of time
- Ability to write clear bug reports quickly and concisely
- Capable of establishing a good rapport with developers and other team members
- A history of amazing coworkers with your bug-finding wizardry

And here's the technical stuff:

- At least basic knowledge of CSS and HTML.
- Bonus points: Ability to write code using C# or similar language (such as Java).

We're also a very social bunch at Campaign Monitor, so you'll need to have a fun personality with great communication skills and be prepared to accept a ping pong challenge at a moment's notice.

To apply, please send the following items to xxx@xxxxx.com:

- Your resume
- A cover letter
- A document outlining how you would go about testing one of our recently-released features, Worldview



Trish Khoo is a Test Engineer at Google, working on Wildfire - Google's social media marketing platform. Prior to joining Google, she created test automation solutions for companies such as Campaign Monitor, Salmat, Ticketek and Microsoft. When she's not doing her day job, she talks about testing on her podcast, she maintains a blog, she writes articles for testing magazines and she occasionally speaks at conferences and community meetups.

Learning To Test With Risk

- Mike Talks

"How do you teach someone to test", is without doubt a difficult question, and the subject of the KWST3 New Zealand peer conference in July. series of pre-written recipies. To teach testing we need activities which are as hands on as possible, and as much as possible exploratory in nature.

It's tricky, we can talk about planning and strategy, but at it's heart, testing is an activity. We can teach common heuristics and methods, but neither is testing just a As a teacher of science, we were encouraged by the UK syllabus to develop "scientific test skills". We would tell our pupils "I want to investigate how weight affects the



speed of how quickly an object falls due to gravity". From this simple directive, students would have to make up their own experiments, collect results, and present findings. This is the heart of science, and the heart too of exploratory testing.

For KWST3 then, I knew I needed to shake things up a bit, and bring something dramatic and visual into the conference. *Because hands on and visual is the heart of testing ...*

Back in 2003 I'd taken up a home woodworking project to build my own bed base. It was interesting, because I took a robust engineers mindset to create something durable – and ended up with something both ugly and incredibly heavy (though functional). I had a lot of leftover material (oops – estimation), and being a fan of Lord Of The Rings, tried my hand at creating my own shield from the leftover pieces. It was based roughly on a Viking shield, although I just made up the design as I went, hammering it together, slats of wood nailed together to a sturdier and thicker wooden spine.



Like my bed design, it's heavy, solid, and it feels really like it could repel anything.

It's not secret that both myself and my son are fascinated by history, although we're drawn to different periods. During the course of a week, we'd watched two documentaries which revolved around shields – one covered the battle of Thermopylae, where Sparta had held a mountain pass from a vastly superior Persian army, and an important factor had been their shields. The other covered "barbarian" armament against the Roman em-

pire, particularly the Francisca throwing axe, which showed one completely shattering a shield of the time.

So this led us back to my shield I'd created, and the question "how effective would it be"? [Notice how general the question is...] We have a wood fire at home, so the house is full of an array of axes of different purpose. I knew by having this question answered, there was a good chance that I'd lose my beloved shield, *but what the heck*. I am a tester, and curiosity got the better of me ... and my son.

We have an axe called a maul at home, it's pretty much what you'd get if you crossed a heavy axe and a sledge hammer.



We've seen our maul axe split giant rings in a single (very powerful) blow. Our expectation was clearly that it would decimate the shield in one blow. So it was obvious that starting with this axe

would give us the answer "against an angry woodsman – you're doomed". But we only had the one shield, so starting there, our inquiry would also finish there, with more questions than answers.

Originally we were going to lean the shield against a prop and hit it (well away from bystanders). But having seen the re-enactments where someone had held a shield and been fine, I felt comfortable about taking a bit of a risk. As an ex-teacher of science, I was trained to evaluate risk – the shield had special extra protection in the part you put your arm, we also wore safety glasses, and I wore a helmet. But I would hold the shield, and my teenage son would hit the shield – *it's one thing for me to risk myself and a trip to hospital, quite another to put another in a dangerous situation*.

We started off with just him hitting it with a common hammer. We weren't sure if the wood would just split.

At this point from our documentary work, we expected that when the shield failed, the wooden slats would just shatter (*an expectation we'll revisit later*).

The hammer just clunked off the shield, putting the tiniest of dents into it.



So we found ourselves a camping axe.

This is a short handled, light axe for use with just the one hand. It's used for chopping wood into kindling sized splinters ...

Again, when hit with this, although there was some recoil and a few dents, the shield could take a continuous number of hits from this weapon.

This led us to our chopping axe, which is twice the weight of a camping axe, has a longer shaft and is a two-handed instrument – *it's the axe type of axe that in the spectrum of wood axes sits just below the maul we were thinking of starting with ...*



This is where it got nerve wracking. He hit it once, and the shield seemed okay. We tried again, and noticed something interesting. The wooden slats weren't shattering ... but ... the impact from such a weight was causing the wood to buckle, and pull away from the spine, with the nails coming severely loose. Against such a weapon, the shield could handle 1-3 hits before the slats would just fall away. But the slats were not shattering at all.

The damage was easy to nail back into place, and I gave my son the shield to hold, as I hit it with a similar weight of hammer. He was fascinated. He knew the shield protected him from penetration from edged weapons, but what up until now he'd not appreciated was that although the shield did protect him from harm, there was considerable recoil from any impact which carried through to him. He thought from watching films and re-enactments that the shield absorbed all this force, instead of passing it onto the holder.

The chopping axe was the limit of our comfort zone – we saw how fallible the shield was against this level of force. Holding the shield and being hit by a maul was just way too risky.

We were prepared to take it to the next level, and bring in the maul. But this time, we put the shield safely on the floor (with no-one holding it), and hit it. To our surprise, the slats still did not shatter, but almost immediately fell away from the spine, almost levered out of place by the force of impact. However even this did not render the shield immediately ineffective, it just created gaps in the pattern.

But we still weren't done – there were more questions. Looking at the shield, the most effective weapon was a two-handed axe like the maul or the chopping axe. But it would take 2-3 powerful hits to remove this from an opponent. It would in “gamer talk” essentially give you 2-3 “lives”.

But to attack with such a two handed weapon, you would have to pull back to strike forward with strength – we ran a few attack scenarios and found someone doing so against a shielded foe was extremely vulnerable to any kind of thrust from the shield holder with either a spear or short Roman sword. Although the giant axe seemed a more powerful and glamorous weapon, the

warrior with shield and spear had the better odds of survival.

We sought to answer a single question *“how good is our shield”*. In actual fact, the information we got was much more far reaching than that, we kept reviewing *“what next”* adding more tests, including acting out axe-man vs shield-and-spear-man. We took risks, but we were sensible about them, and we reviewed them. We made the call after the chopping axe showed signs of failure, to change tactics for the next level of test. We found our shield behaved in ways we just could not have predicted, even from documentaries ... it actually turned out from investigation that our wooden slats were twice as thick as those used in Viking shields.

It also leads to more questions. Would thinner slats fail just from a camping axe? If so, how much good would they really be? Does using nails or screws to fix the slats in place make a difference? Questions we're thinking of trying to build mock ups to investigate over the New Zealand Summer. This is the legacy of the Mythbusters generation, and perhaps there is no better program to entertain and teach approaches to exploring questions and discovering knowledge ... this is testing at it's rawest – the curious search for *“what if's”*.

This demonstration and theme formed the core of my experience report at KWST3 ... believe me, being hit by an axe first thing in a morning at a peer conference is a great way of asking your audience *“are we awake now?”*



Mike Talks is a historian, wargammer and part-time lumberjack. Although not originally a New Zealander, Mike Talks feels that Wellington is one of the most

exciting places in the world to be working in software testing right now. He works for Datacom in Wellington, where he is training an army of Spartan testers.

He is the author of the Leanpub book on testing *“The Software Minefield”*.

Are You Listening to Your Intuition?

- Bernice Niel Ruhland

As Software Testers we spend time writing test plans, identifying and addressing risks, and writing test ideas or test cases. Through testing we learn more about the feature we are testing and potential risks. We may change directions removing tests we thought we would perform in favor of ones that address new risks. We attend training seminars; discuss testing approaches with other testers; and read blogs, articles, and books. To further our critical thinking and strategic skills, we may participate in weekend testing, perform testing challenges, or play games that require us to change our strategies based upon what we are learning. We invest a lot into our software testing careers to further our knowledge and abilities.

A lot has been written on how to progress your testing skills but an area often overlooked is listening to our intuition. The Free Dictionary defines intuition as: The act or faculty of knowing or sensing without the use of rational processes; immediate cognition. A sense of something not evident or deducible; an impression. Sometimes during testing something does not feel right to us or we might feel something is missing. I have found that when I do not pay attention to my intuition that I may miss something important. This can be related to my own testing, when reviewing testing approaches written by other testers, or when I am reviewing test results.

When your intuition tells you something is not right or something is being missed, take some time to investigate. This can be at any stage of testing such as: test planning, test review, or test execution. I believe a Software Tester's intuition is based upon factors such as testing experience, knowledge of the product, and previous bugs encountered in that product area or similar type of coding. Below are a few tips on what you might do to get started. Because we can easily go down rabbit holes during these hunts, it is best to time-box any activities.

- Skim through any strategic documents, test plans, and requirement documents.

-- I have often found something in the test planning document that has been overlooked by the testing team once they progressed to deep testing. Ideally test plans should be kept lean and evolve as you learn more through testing.

-- For requirement documents, I often like to skim the table of contents and major headings to see if there is anything of interest.

- Has this functionality been tested previously? What bugs were encountered? What testing strategy was used?

-- Often when we review prior test documentation we find something that helps us make better testing decisions. Did the code have any serious bugs during a

previous testing cycle - those might be good areas to retest.

-- Can we reuse or modify tests previous created?

- Is this new functionality? Is it similar to another area of the product? Are you seeing any testing patterns similar to that area? What was learned testing that portion of functionality?

--We can learn a lot by how we tested similar functionality to define testing approaches and sometimes better understand potential risks. Reviewing test documentation for those areas can be helpful in identifying starting points or additional tests.



Bernice Niel Ruhland is a Software Testing Manager with more than 20-years experience in testing strategies and execution, developing testing frameworks, performing data validation, and financial

programming. She uses social media to connect with other testers to understand the testing approaches adopted by them to challenge her own testing skills and approaches.

When not exploring the testing world, Bernice enjoys cooking and spending time with her husband living a health-conscious lifestyle. The opinions of this article are her own and not reflective of the company she is employed. Apart from other activities she regularly contributes to Testing Circus Magazine. Bernice can be reached at:

LinkedIn:

<http://www.linkedin.com/in/bernicenielruhland>

Twitter: bruhland2000

G+ and Facebook: Bernice Niel Ruhland

Autobiography of Automation

- Yagnesh Shah

Once upon a time in the magical era of testing, I was born with my charms of efficiency & speed. I got real famous due to my good qualities of producing fast results & slowly I began to replace human efforts into scripts. One thing anybody can agree with regards to modern era is that news spreads like Virus. My good qualities became latest trend on Twitter, My blogs & characteristics got over millions of Facebook Share, People started digging me on Digg, I received many +1 in google plus & the tales went on & on...

Any guess who am I?
I am "Automation... Check Automation!"

I started to get job offers from across the world with handsome packages. My concept & ideology got converted into various tools like [Sahi](#), Selenium WebDriver (JAVA, PHP, Python), Twist, AutoIt, OpKey, Cucumber, Robotium, SoapUI, QTP, etc. My qualities got matured over the period of time, I could do record & play directly by interacting with GUI instead of writing scripts & commands & many other cool features. This is a busy world & time is being considered as money. People started working along with me to make record & play scripts along with some manual customizations. There were times when scripts are written to automate cool scenarios which adds value to the project & client.

People working with me started making scripts for ready-made test scenarios which were manually being tested for Regression Testing. It is great to see that my concepts are helping out everyone to reduce their workload. Over a time, I was started to be mistaken as Automation & as I mentioned, you know that news

spreads like Virus. ;) There are variety of aspects which were being overlooked in terms of what Automation means:

- My cool friends who are good Automator & tries to make cool scripts are being given loads of ready made test cases & scenarios to Automate in order to meet charming deadlines.
- My friends are unable to get chance for thinking & designing cool test scenarios which got missed in ready made test cases document for Automation.
- My Friends may not find enough time to perform Blended Automation Testing.
- My Friends may find it difficult to spend time to work on extra activities & indirectly add value to the company.

Example: By creating their own tools & scripts to make testing for everyone easy & fast which in turn allows everyone to spend time on actual Brainual Testing.

Gradually my ideologies & understandings started to fade. People are racing with everyone to be the number 1 in the market. Success is sure to visit your doorstep by being the number 1 in the market & meeting deadlines. But, being the number 1 in adding more value to the Project/Client/Company indirectly makes you more popular, both within 4 walls of your company & beyond that, which is way cool then being only successful.

"As one person I cannot change the world, but I can change the world of one person" ~ Paul Shane Spear

I am not trying to set any new trend by setting my nickname as “Check Automation!” or “Test Automation!” or blah blah...

Purpose is not to confuse everyone with nick names of Automation (like [Black Viper Technique](#)), but to set the mindset of what I want to be famous for & what value can Automation add.

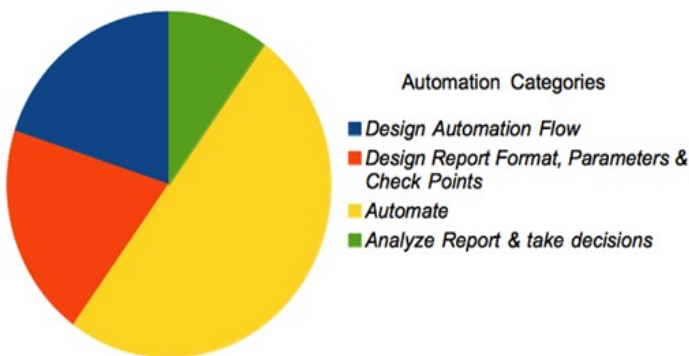


Part 1: What I Want To Be Famous For

As a way of gathering information which can be accessed by variety of audience (like Business, PM’s, Developers, Testers, Users) & appropriate actions can be taken in a much much much faster way.

Word of Caution: Do not fall into the trap of considering this as Bible for Automation definition. It’s just a thought & it holds the scope of maturing over the time.

Generally every company has a [Automation Framework](#) to be implemented for any Automation project which reflects the credibility & efficiency of Automation scripts. Few simple steps for completing Automation scenario can be categorized in 4 steps.



Example:

- **Scenario:** For any eCommerce site, “Feedback form” is a critical process & we need to make sure that for every build it should be **working as expected**.

Step 1 - Design Automation Flow:

“Working as expected” is an interesting statement highlighted in bold for above statement. What does it mean? We can categorize features & functionality of “Feedback

form” into different ideas & a complete scenario can be designed for Automation flow.

- Following are such categories for “Feedback Form”:
 - Presence of Feedback form across the site like Homepage, Product Page, Contact Us Page, e-Gift Voucher Page, etc.
 - Appropriate error handling message presence
 - Successful form submission
 - Checking for different categories of feedback form like “Improve this page”, “Suggest new features/ideas”, “Shopping experience” & other categories.
- **Benefits & Necessity:**
 - Automation Testers are getting opportunity to think as a Context Driven Tester & design the flow instead of following ready made steps from test case document.
 - These phenomenon can be referred as **Blended Automation – Combination of Automation & Testing**.
 - Make a list of “Value addition” being done by each scenarios. It assists in highlighting the importance of each scenario to the variety of audience.

Step 2 - Design Report Format, Parameters & Check Points:

Once the flow is designed then comes the turn for designing the report parameters & check points which needs to be considered for reviewing by variety of audience.

- There are lot of ways & tools like TestNG, Maven, TestRail & others to create reports. But all they do is show some colorful graphs & Statistics on % of Pass, Fail, Error, Warnings.
 - Sample Reports:
 1. [Sahi - Reports with graphs](#)
 2. [TestNg with Ant Report Overview](#)
 3. [TestNG with Ant Report Detailed](#)
 4. [Maven Report](#)

- 5. Ways of Sample reporting can go on & on... Everything depends on the context & information you wish to highlight.
- We need to think of few other parameters or customized messages for report logs which assists in providing a brief insight to the behavior. Such information can actually assist in tracking down where exactly is the issue noted.
 - Sample Customized Report: Feedback Form customized Report - <http://screencast.com/t/tbaup4fFmBVb>
- **Benefits & Necessity:**
 - This is critical, since “Automation Report” is the medium of conveying all valued information.
 - *It also assists in making necessary decisions such as “Feedback form is reliable for the current build & we can focus on making sure other features of the build are good”.*

Step 3 – Automate:

It's time to make a script for the desired flow by keeping the context in mind.

- *Make sure to note any absurd behaviors in between the flow & try to report the same.*
- Example - “Reporting any absurd behavior”:
 - For instance, login is sometimes taking more than 4-8 seconds. We need to report such instances in our Report logs since such information may be critical for evaluation & impacts the business.

Step 4 – Analyze Reports & make decisions:

Doing so helps business to make sure that

- “Feedback form” is working as expected & focus on other modules of the site
- Revenue is not being impacted
- We can get a pool of information for site improvements, suggestions, user review, etc.

Part 2: What “Value” can Automation Add



- **Blended Automation**

- My friends can get opportunity to think as a Context Driven Tester, design the flow & observe while writing scripts.
- It is not boring & keeps the curiosity alive since you are not following blind instructions from test case documents.
- Being an Automator, it is damn difficult to get time for polishing testing skills & making test ideas or scenarios.
- Unless you become a good tester & observer, it will be difficult to analyze what information to log for reviewing.
- Refer example of “Reporting Absurd behavior” in “Step 3 - Automate” topic under Part 1.

- **Replacing human...**

- Oops! A small correction. It's “Reducing human effort... Not replacing humans” :)
- My concepts & thoughts are meant for reducing human efforts & assist them in any way possible to collate information in a much faster way.
- Highlighting such critical information is the sole purpose I exist.

- **Utilize Add-ons & Tools available**

I am happy to see my friends sharing & utilizing Tools, Scripts, Add-on's which are available in the market & not to duplicate the work by making scripts for the same.

- Example: There are lots of link checkers & tools available to make sure the HTTP Status of Links across the site.
 - Xenu
 - Integrity
 - Link Checker Add-on
 - Check My Link Add-on
 - and so on...
- Utilize the resources available & save time for other cool Brainual Testing & Automation.

- **Make Tools & Share**

Probably you have reached that rebellious age

- where no more functionality Automation can be implemented for the site or

- You have exhausted time & resources for Automation.

Congratulations on successfully completing a project. It's now time for sharing your experience & challenges. Based on your observations if you think that you can assist developers or team by making scripts for their day to day unit testing then go ahead. It is a critical step in adding more value to the team apart from your cool Automation scenarios for testing.

Example:

- For a Media site, it is critical to make sure that images are being loaded across the site.
- As a Context Driven Tester & the availability of time, we can make sure that images are being loaded across different categories of the site like different content types (Articles, Slideshows, Editor Pages, Recipes, etc).
- But what we are achieving is only limited information based on limited time & efforts that "Image Loading" is reliable across the site. This is kind of "QA" we did, instead of "Testing".
- "QA" - because we are making sure that quality of "Image getting loaded" is maintained across the site & the functionality works without breaking in the current build. And hence the build is good for release or not.
- "Testing" - How to do Testing for such scenarios which impacts the revenue, users & traffic of the site.
- We can make a script (Tool) to perform Unit Testing of "Image getting Loaded":
 - Collect a list of URL's from the site which contains Images
 - Open each URL at a time & extract the names of all Images & store it in an excel sheet.
 - Now, make sure all the Images exists in the image server by cross checking the names from the excel sheet.
- Doing so enables anyone to run the script for any number of given URL's & make sure appropriate information is highlighted in Report Logs for reviewing.

- Such activities can be considered as Testing instead of QA, since we are highlighting information based on much wider coverage & scope which impacts the business.

It's a great feeling sharing my Autobiography with you all. Especially thoughts on "What I want to be famous for" & "What Value can Automation add". See you all ...Until Next Time...



Yagnesh Shah started his career as a Tester & was not sure if this is what he wanted to do for the

rest of his life. Within 1 month of being in the industry, he figured out that there is a lot to learn & share in this domain. He acknowledges all the guidance received by his colleagues at Moolya for all the learnings & for providing a platform to experiment his ideas with freedom. He believes, he is the Gen-Next Tester & it is his responsibility to share the same with the rest of the world. He is an avid blogger, sharing his thoughts @ <http://yagnesh23.wordpress.com/>. He loves to talk about Testing and highlighting information as per the context. He has a keen interest towards understanding Automation. His experiments on Automation in Moolya (with guidance from colleagues) led to creating pillars for Automation Group (called Transformers) & has been able to make an Automation Framework for Moolya.

Heuristics

– User Experience Testing

- Dheeraj Karanam

User Experience (UX) is a user's perception of the practical aspects of the application, product or system like ease of use, efficiency, etc. It includes all the user's emotions, physical and psychological responses, preferences, beliefs, behaviors and accomplishments that occur before, during and after use.

While testing an application, we consider various quality criteria. Every aspect has its own flavor and adds up to the taste. User Experience is one among the need-to-consider aspect of testing and a powerful aspect that claims a large share in the success of the application.

Your product may be functionally unbeatable or marketed largely but if it does not satisfy or enhances the user's experience, caution – you may be anytime heading towards downhill. If you want to lock and retain your customer, the powerful tool is to give the best user experience. It's after all the users who make the success or failure. So it's become important to pay more attention to this

Why is User Experience testing important?

Scenario 1: You are hungry. You go to a restaurant, order your favorite food. What if it is served in an untidy plate?

Scenario 2: You need to travel and reach the office on time. You caught the bus at right time and it is going at a appropriate speed. What if the bus is crowded?

Oh crap!

This is what we say when we have a bad experience. The question here is, why are you not satisfied, even when your purpose is met? There must be something

else that bothers you and keeps you unsatisfied, though the requirements are fulfilled.

In above scenarios, explicit requirements of users are clearly met. Implicit requirements which were never on paper but were always in the minds of users were not met. While testing an application, it is necessary to test for implicit requirements as well. Sadly, it is not as easy as showing a score card and putting your rating on it.

Heuristics

'Heuristic is a mechanism to find a bug, which may or may not be fruitful'

The Fundamental aspect of any product is to 'Satisfy the User'. This way, every heuristic we use while testing adds up to User Experience testing.

When we talk about user experience – not all fingers are same. Users vary largely. I have come up with a Mnemonic – A-REC B- REC that testers can use while testing for User Experience:

Audience

While testing we need to test the app keeping in mind the type of audience who are going to use it. Different audience looks for different things in the app. For example what a kid expects is not the same as what an old man needs. So testing with the mindset of right audience will help a lot. This is also called as User Persona. It is important to come up with different User Personas based on age groups, tech-savviness and other factors.

Relevancy

User shouldn't witness any irrelevant messages in the application. For e.g., the error messages shown should be relevant with the error occurred. The User should be able to understand the error on reading the message displayed. Error messages must be good enough to explain what went wrong, direct the user to next right action and restore the application to a stable state.

Emotion

Human emotions play an important role to decide the User Experience of the product. It is all about how the emotions of a user change while using the application. There may be few aspects of the application those turn off user's mood, for e.g., If a page takes more time to load, the user may be pissed off. We might have encountered many such situations and quit looking at some applications. We can recall or conduct survey of such experiences and test for them.

Competitor Analysis

In this competitive world, you can stay ahead of your competitor by providing better User Experience. Considering the User Experience with other competitive applications, we can end up with a bunch of ideas to test upon. Looking at the competitors work, we can understand different aspects of the product and come up with a list of ideas that could help our product to look better, serve better and experience better.

Branding

Though the branding is always essential in any application, it shouldn't dominate the content and annoy the user. Also, we need to make sure enough branding is done. There is a saying 'Perhaps too much of everything is as bad as too little'. So, it is our duty to understand the context and suggest the amount of branding to be done.

Reviews

Considering various reviews given by the users for similar products, we can come up with test ideas that could help development team to fix important problems quickly. Reviews act as a good mode to scale the User Experience of the application. Social networking sites, public forums act as good review sources, whereas they need not be reliable as well. It's our duty to review the reviews and come up with our version of it to generate more test ideas.

Expectation

User expectations are the deciding factors for the success of a product. Keeping in mind the brand of the application, kind of application, success of previous releases of the application, users might be having some expectations. If the application doesn't meet user expectations it may turn out to be a failure.

Content

The content of the application should be relevant and appropriate to the customer's needs. It also should be according to the business goals. The content should be structured in such a way that the user has a good experience while reading it. The content should be appropriate enough that the users find what they want with utmost ease when they are in the application.

While implementing these heuristics, we assume the role of a user. We end up focusing on behavior that we as testers expect rather than what other users expect. We need to make an attempt to balance this bias and find ways to identify user patterns that are very close to real life scenarios.

Do you have a user experience story to share?



Dheeraj Karanam, has started his journey as a Software Tester one year ago. He works as an Exploratory Software Tester in Moolya. He strongly believes that passion, planning and intelligent hard work would bring in success and has been practicing it. He shares his testing time stories through his blog - <http://bloodytester.blogspot.in/>.

Apart from testing, he is occupied with social service and has co-founded a charitable society Lakshya-Changing Lives. He is big time foodie and enjoys music. He is tweets at @mindseye04.



- Nagasahas Dasa

Security Testing for Beginners

There is no wrong way to start hacking, everything is right way and I have my own way. Whatever your style of hacking is, make sure it's consistent. If you are starting out today you can be benefited based on your skill sets. Don't learn to hack, hack to learn.

Well, coming to the point how did I start hacking or how did I land up here, It was in the year 2008. I was in my 2nd year diploma where one of my friends was trying to download videos by searching on Google. In 2008, getting a video to your local machine was one of the biggest achievements for people of my age. My friend showed me how to get the videos from Google, by extracting only videos from the vast search results.

He asked me to enter some string along with the search query.

Filetype: avi <Search Query>

He didn't know what it was, and he told that he came to know about it through his senior. Ok!! As I am very much interested in computer technologies, I tried to find out what they are. I referred to many articles and found that they are called as **GOOGLE DORKS**. I even came across some of the terminologies like White, Black and Grey hat hackers. During this phase, I got a common response from whoever I asked about hacking, which was "Hacking is very difficult and I don't know anything on it except that it is illegal".

But, it is not illegal as I told you before. There are 3 categories of hackers:

- Black Hat Hackers
- White Hat Hackers
- Grey Hat Hackers

Black Hat hackers are those who perform undercover hacking for malicious reasons and also with intent to harm others, such people can also be referred to as 'crackers'.

White Hat hackers are those who perform hacking for legitimate reasons and use their skills and knowledge for good, e.g. IT Security technicians testing their systems and researchers testing the limitations of any software.

Grey hat hacker is a combination of a black hat and a white hat hacker. A grey hat hacker may surf the internet and hack into a computer system for the sole purpose of notifying the administrator that their system has a security defect.

According to a survey the most common technique of hacking a website is SQL Injection. SQL Injection is a technique in which hacker insert SQL codes into web form to get Sensitive Information like (User Name, Passwords) to access the site and deface it. The traditional SQL injection method is quite difficult, but nowadays there are many tools available online through which any script kiddie can use SQL Injection to deface a website. Because of these tools, websites have become more vulnerable to these types of attacks. Some of the tools used for SQL Injection are mentioned in this article. However, as I know nothing is bug free and there will be exploits every minute/hour.

Some of the tools which help in finding the vulnerabilities are discussed below:

1. **Wireshark** is also known as Ethereal. It is one of the most powerful tools in a network security, as a network packet analyzer on any network. It is used to capture each packet sent to or from your system to the router. If you're capturing on a wireless interface and have promiscuous mode (Admin/super user) enabled in your capture options, you'll also see other packets on the network sent from different nodes. This also includes filters ex: DNS, TCP, UDP, ip.addr etc), color-coding,

capturing packets and other features that let you dig deep into network traffic. Wireshark is an extremely powerful tool; this is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals. To get this position, it takes a fair amount of practice. It takes practice to know how and where to capture right data, filters to use, and how to interpret the data.

People willing to learn can use this link to get sample captures on Wireshark to get experience hands on this <http://wiki.wireshark.org/SampleCaptures>

2. **Fiddler** is an open source web debugging tool which captures all the traffic between your computer and the internet, it also acts as proxy between the browser or any application on the local machine and the internet say, all the traffic flows through the fiddler and the requests can be altered and the altered request is been sent to the server. In simple words fiddler sits between HTTP client that is the browser and the HTTP server.

Normally it would be configured with all the browsers being used on a particular machine or you may have to manually configure the browser to capture all the traffic in/out of our machine.

Fiddler can also be used to find the statistics, inspect the request or the response and can even act as an auto responder and is capable of sending request from the fiddler wit out any browser. Fiddler is designed in such a way that it capability to run API's through composer functionality and can even right some scripts which can be helpful for check automation and has the capability to decrypt HTTPS traffic.

3. **Nessus**, the first public release was in 1998. Nessus was an open source vulnerability scanner, recently nessus turned into a paid tool. This tool is used for scanning both web application and network, Network can be either internal or external IP/Network. Nessus is designed to automate the testing and discovery of known security problems. Allowing system administrators to correct problems before they are exploited.

Nessus uses a client server design that allows the user to set up one server that has multiple nessus clients that can attach and initiate vulnerability scans, where servers can be placed at various strategic points on a network allowing tests to be conducted from various points of view.

Nessus security checks vulnerabilities and database is updated on a daily basis which could be retrieved to cross check the database with the command "nessus-update-plugins".

4. **IBM Rational AppScan** is an automated web vulnerability scanner which helps in finding the vulnerabilities quickly and effectively, even a svan (semi technical person) can also use the tool and find vulnerabilities. Using IBM app scan, we can decrease the risks in web application attacks and data breaches. It helps in testing the web application either on production site or on any staging sites which can ensure that it checks for web attacks.

Basically in IBM AppScan once you add a web app to test for its security the initial step is to crawl all the pages/links on that application which are allowed to be crawled based on robots.txt

Basic functionalities of IBM AppScan are

1. Gives the larger coverage of test report
2. It mainly concentrates on top 10 OWASP (Open Web Application Security Project) web application vulnerabilities.
3. Accurate and advanced scanning algorithms used hence less false positives
4. Recommendations, Which I personally like here, It gives us description of each vulnerability found and the risk involved in not fixing it.

As we all know automated scanning is not perfect all the time and is not advisable to completely depend on automated scanner, hence they have provided a manual scanning for any vulnerability found to give the perfect solution without false positives.

IBM app scan is a paid tool and it has a trial version as well if you are interested in exploring the application.

5. **Nmap**, also known as "network mapper", it is an open source application which helps in quickly scanning different ranges of devices such as desktops/laptops or any mobile devices and provide valuable information about the devices which are connected to a particular network. Nmap is available for all the platforms where it can be operated in 2 ways, command mode and GUI mode but most people prefer command mode for its advanced features but requires technical knowledge.

Nmap uses raw IP packets to determine what hosts are available on the network (Host Detection), the services that are enabled, the operating system and version, using TCP SYN or a TCP Connect ping to gather active hosts. Nmap is used by security researchers and hackers who want to find the weakness and exploit them.

Nmap can provide different types of scans, where some are more aggressive and some are simple, designed to be stealthy and scan undetected. Depending on the type of scan performed, different information can be discovered; some of the scans are Ping, SYN Stealth, UDP Scan, IP Protocol Scan, ACK Scan, RPC Scan, List Scan etc.

6. **Havij** is an automated SQL Injection tool that helps hackers or security researchers to find and exploit SQL Injection vulnerabilities on a web page on a vulnerable web application, using Havij user can access database, retrieve DBMS users and password, dump tables and columns, fetching data from the database, running SQL statements and executing commands on the operating system.

Hackers use Havij along with vulnerability scanners such as IBM AppScan or Web Inspect, vulnerability scanners find vulnerabilities but not help you in actual exploitation and that's where Havij showcases its functionality. In other words, vulnerability scanners will help you in finding list of vulnerable webpage's whereas; Havij helps you with the access to the database for entire exploitation.

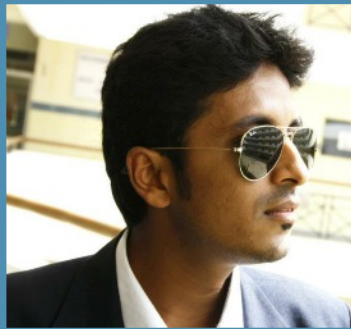
Once URL is feed to the Havij, it comes up with a list of databases being used, version, and db-name's. Later selecting a particular database we can drill down to tables, and then to columns and even to the actual data. Passwords would hashed usually, there are set of decrypter's associated with the tool which help user to

decrypt the hashed password, it is also associated with an algorithm which helps users to find the admin page of a particular web application. In simple words it's more useful for hackers than security researchers.

7. **SQLMap** is one of the most popular and powerful open source SQL injection automation tool, which is built on python and can run on any platform if python is installed in it.

Giving a vulnerable URL, SQLMap can exploit the database and provides with sensitive information like extracting database names, tables, columns, all the data in the tables etc. It can even read and write files on the remote file system under certain conditions.

We can run this application only on command mode and doesn't have an interface, and has simple commands to extract information from the database.



Nagasahas Dasa is a young Information Security freak who loves to explore & learn more about technology and love testing. He works for Moolya Software Testing Private Limited as an

Exploratory Software Tester. He is a major contributor for Mozilla and have been recognized for his skill sets on security, even got a bug bounty for one of the security bugs he had reported. He has given various talks on information security in public to spread the awareness of security and hacking.

Nagasahas alias Sahas is not just passionate about his work but also crazy! He makes things happen with great interest and love. His expertise in testing is not just limited to security but also functionality, usability, user experience, SEO and much more. In simple words he is in love with technology.

You can follow him on twitter @nagasahas. He blogs at <http://solidmonster.wordpress.com> and you find him easily on Google or Facebook as he is famous for his achievements and visibility.



...Research... ...ReCycle... ...Reseed...

- Trinadh Kumar Bonam & Indrani Areti

50 years ago people in the high end of the society used to witness different cultures only when they visited different countries as tourists. As a result of globalization, international business grew rapidly after the beginning of the 20th century. Economy, technology and man power are the main reasons for globalization. Exchange of world views, products, ideas and other aspects of culture increased drastically.

In the days where telegrams, telephone, videoconferencing, intranets, Internet and email are not there teams need to be in the same physical location in order to work effectively. Work locations are moved from different building to cities, then to countries, and even continents. Team members may be in different time zones, speak different languages, and be part of different cultures.

Different sources such as Offshoring, Outsourcing, crowd-sourcing and near-sourcing are playing key role in distributed team management

Offshoring: The practice of basing some of a company's processes or services overseas, so as to take advantage of lower costs.

Outsourcing: Obtain (goods or a service) from an outside or foreign supplier, especially in place of an internal source.

Crowdsourcing: Obtain (information or input into a particular task or project) by enlisting the services of a number of people, either paid or unpaid, typically via the Internet

Near-sourcing: Near-sourcing is a term used to describe a business strategically placing some of all of its operations close to where its end-products are sold

Software testing is one of the fastest growing segments within the IT services industry. On an average, software services are growing at about 10-12 percent, testing is growing at more than 50 percent per annum. Leading IT companies in India today derive up to 10 percent of their revenues from their independent software testing businesses.

Why to offshore Testing?

Some of the reasons are listed below

- a) To decrease dependency on one supplier
- b) To increase competition between suppliers, ultimately getting ROI in the form of Quality
- c) To decrease risk
- d) To get access to dedicated service providers who have solid expertise in testing practices
- e) To reduce operating costs
- f) To reduce time to market
- g) To achieve continuous productivity (*by handshake mechanism between dev and test teams*)

Consider different outsourcing options, from bringing in external testing experts to setting up an offshore development center, to find the most cost-effective alternative. Whatever it could be ultimately the teams are distributed. Moving testing activities to a different location from development team is risky.

Offshoring is seen by many as introducing more problems than it solves. Even companies with less than 10 employees are also having distributed team in more than one location. It is more difficult to systematize and to develop sound methodologies and processes in a distributed test team. Projects that are globally distributed are having different logical properties like **Languages, Cultures, Customs, Attitudes etc.**, are majorly impacting the success of the project. These may or may not be show stoppers in distributed team management, but we require a recycled approach to fill the gaps in project coordination, people management and communication.

In this article you are going to notice how we can overcome these limitations by using an approach 'R³ ReCycle'. R³ Recycle approach will identify and resolve the issues in the logical properties, of the distributed team. We have 3 step processes in Identifying, Eliminating and Streamlining different glitches in distributed team management especially when a test team is at offshore. Three different language concepts are used to remove the glitches in the three logical properties respectively.

Since communication being the strong pillar in a distributed team we concluded by throwing some light on different tools that are being used or suggested.

Before we see the 'R³ Recycle' approach usher you into its conceptual language; let us steer to examine how and what are the different challenges test managers faced in a distributed team management system and the various dynamics that impact or influence globally distributed teams.



Different challenges in a distributed test team management

a) Managerial

This is one of the key challenges all the test manager(s) face in handling clients, stakeholders and internal/external teams. Leadership and management from a distance, depends on invisible element called "TRUST", though it may not work all the time. Bonding isolated teams and building business relations to synchronize the team members is more challenging with globally scattered teams than the co-located teams. Management needs a striking balance in controlling the dispersed team(s) in making

the team members feel comfortable to report problems and to raise opinions openly. Utilizing the available resources accurately to avoid duplicate efforts and promoting personal accountability is a leadership challenge.

- Lack of clarity regarding roles and responsibilities
- Poorly defined expectations
- Poorly defined deliverable
- Conflicting management direction
- Mismatched practices and processes

Communication



b) Communication

Maintaining communication richness over distances for successful celebration of projects involves exchange of thoughts, sharing of information as by speech, visuals, writing or behavior with all parties of dispersed teams. When testing is spread out in location, communication becomes difficult. It is important that communication is clear, honest and consistent so that everyone gets the same picture. Multiple ownerships for a single activity create lots of communication gaps. Improper or no communication of milestone deliverables creates gap between the core team and management. Setting goals and objectives irrespective of competencies creates conflicting management direction.

- Slow network connectivity
- Software incompatibility
- Available hardware

Technical



c) Technical

The electronic connection plays a prominent role in bridging the gap between technical resources in cross-functional and cross-site teams to help the teams to be on the same page. Compromising on technical services like VPN connectivity, network connectivity, server performance etc., would damage reputation and business.

Here is the checklist for Test Manager(s) and Test Lead(s) which helps them in handling above challenges

Checklist	
Test Manager (Client from Onsite)	Test Lead (Supplier from Offshore)
Ensure proper hand over and takeover of testing activities	Ensure proper hand over and takeover of testing activities
Encourage varied talent for quality output	--
Use common repositories for tracking activities	Use common repositories for tracking activities
--	Follow client terminology though you have a specific one
Maintain transparency	Maintain transparency
Visit offshore team to increasing association and plan for outing.	--
Plan for training if required	--
If possible bring them to onsite on a rotation basis	--
Ensuring new employee smooth on-boarding	Make sure relevant documents are updated for transition
Keep posting the team with latest business expectations	Keep posting the test manager with latest updates in the team
Control your communication	Control your communication
Concentrate on people attitude	Do offline communication regarding people attitude
Make sure one strong functional resource is available while UAT is going on	--
Keep track of prospect future lead to avoid risk	Remove dependency by assigning cross ownerships
Avoid using 'But' in business conversations. Note: I've seen managers using the word 'But' and giving a long gap in conversation.	Never communicate to client something like this "I need all those approvals, without those I can't proceed". Better use 'We', as you are representing a service provider.

Dynamics influencing/impacting distributed test team

Good dynamics start with an effective test manager. It can be done through smart staffing, positive guidance, and fair dispute resolution. Different dynamics that influence distributed test team are:

- a) Productivity and Flexibility
- b) Diversity of talent
- c) Cost benefits and Minimal infrastructure
- d) Work-Life balance and Individual control
- e) Market opportunities and Client satisfaction

a) **Productivity and Flexibility:**

Fully distributed teams going after “Follow the Sun” approach, often see a proportional increase in productivity as the daily remaining/completed work is always handed off between the global teams either it could be test/dev, not limiting the work day schedule to traditional 9-5 hours. Hand over the tested functional areas to onsite dev in order to fix the defects. The team on the other side of the globe simply picks up those defects and works towards resolution.

b) **Diversity of talent:**

Dispersed teams across the globe, due to its nature of workplace diversity, stands out to attract the best available talent and skill in the market both domestically and abroad. Such dispersed teams allow test managers and organizations to tap into new global markets and make best use of globally available talent. These teams having diverse talent and backgrounds enable better utilization of talent pool and benefit from diversity in creative thinking, problem solving, improved team morale, improved quality in the produced deliverables and an increase in visibility of projects.

c) **Cost benefits and Minimal infrastructure:**

In order to reduce costs, transfer risk companies started off-shoring or outsourcing their services to vendors or service providers. Other main reason for outsourcing is to mitigate shortage of skills. Sharing infrastructure and office spaces would reduce the total cost. Scattering the services to different service providers in different geographical locations will increase productivity, competition and reduce risk.

d) **Work-Life balance and Individual control:**

Work-family-balance is considered greatly as an individualized concept where satisfactory balance is different for different people. It makes impossible for many employees to find a satisfactory balance with unpredictable work demands, long working hours, rigid start and finish times. Organizations provide better policies these days for its faculty and staff to balance their responsibilities and interests at work and outside work to aid in the recruitment and retention of talented employees. Employers have also found distinct benefits in making work-life balance possible for their employee by promoting family-friendly work environments, implementing flexible work arrangements and encouraging creative work policies.

e) **Market opportunities and client satisfaction:**

This is a major benefit of geographically dispersed teams due to direct access to different market opportunities. With work teams located and extended in different parts of the globe, organizations are able to establish their presence with clients worldwide. With the ability to compete on a global scale and without being limited to a particular client location, the extended teams resulted in maximizing teams’ performance and revenue growth while further increasing client satisfaction.

R³ ReCycle Approach

The diversity in cultures, languages, customs and attitudes is an obvious feature of a business which is spread vertically and horizontally internationally. Distribution of tasks across borders not only brings out potential benefits to team and organization but also raises political issues that can hinder a team's progress and affect team management and professional relationships. Spatial - Temporal - Cultural - Linguistic differences play a major role in any globally scattered team. It is very important to understand how these factors affect the global teams and how test managers need to take the consequences into account when planning activities and scheduling.

A test manager who is leading a distributed team needs to accommodate differences in the logical properties of a distributed team like culture, working hours, time zones, local conditions and languages. These differences need to be handled carefully in order to retain or gain customer satisfaction and business.

R³ Recycle approach will identify and resolve the issues in the logical properties of the distributed team. We have 3 step processes in Identifying, Eliminating and Streamlining different glitches in distributed team management. The concept of this approach is - as the project progresses; recycle the existing processes on a continuous basis as the team structure is dynamic. Dynamics in the team are in the form of

- a) Resource ramp-up or ramp-down: Hiring new resources from different regions, cultures and attitudes. Hiring resources that don't have domain knowledge.
- b) Day light saving: Time zone changes
- c) Geographical position: Opposite sides of the earth

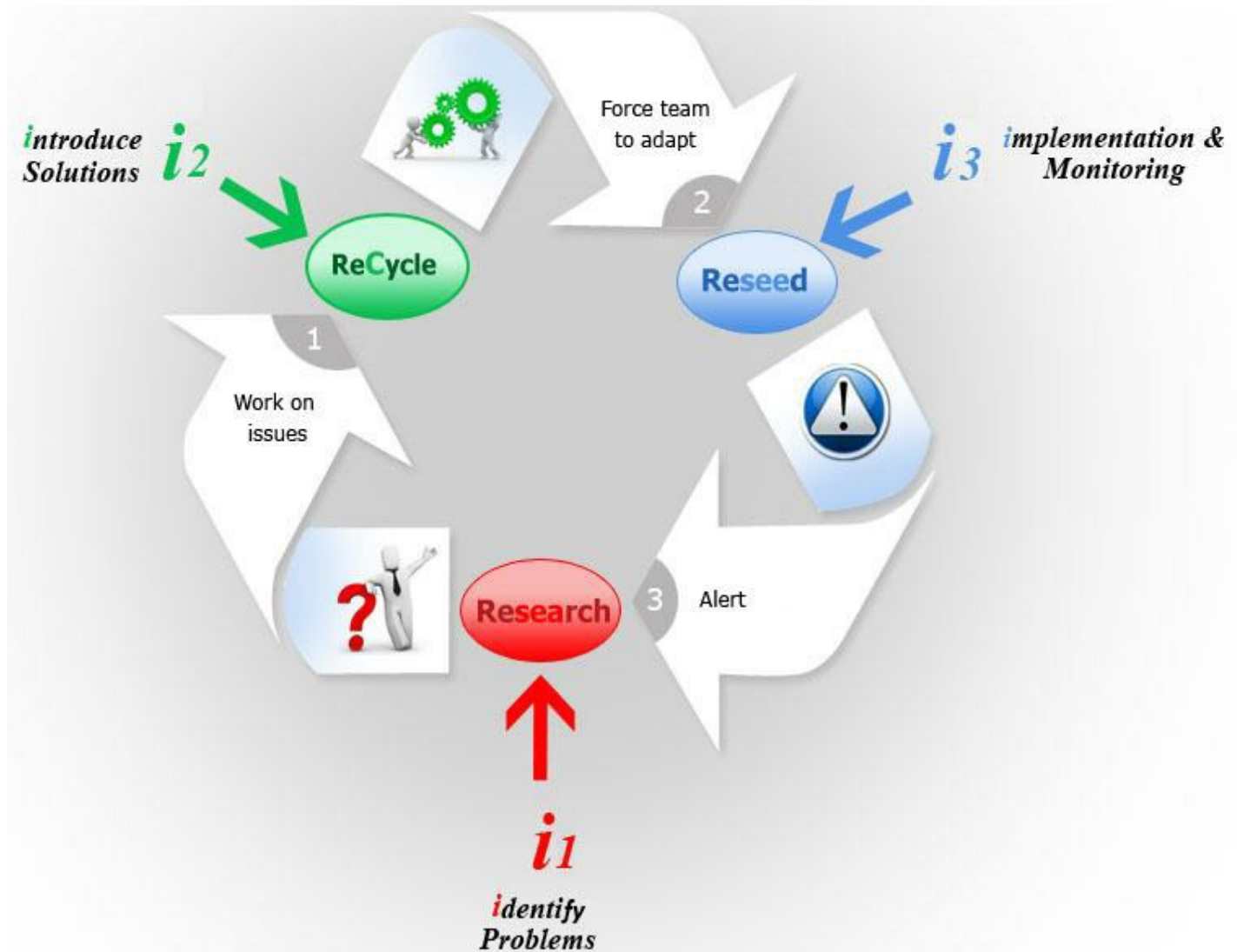


Table 1

Logical Properties	Problem	Resolution
Language	Dissimilarities	Research - search for problems , research on them
Time zone	Variances	Recycle – use the existing principles and bring out new solutions
Geographical & Cultural	Conditions & Differences	Reseed – seed new ways to reset the process

In the below consecutive paragraphs, three different language concepts are used to remove the glitches in the three logical property problems respectively.

Table 2

Logical Property Problem	Language Used
Language dissimilarities	No Sea Language
Time zone variiances	C Language
Geographical conditions & Cultural differences	See Language

Note: In order to elevate our language solution, we colored the language in the image 4 and table 1. Let's see the logical property which is considered as a problem in distributed test team management and the respective language solution.

1. Language Dissimilarities:

It is necessary for all the team members working together to be on the same page at every phase. Though English is chosen medium of communication, the reality of global implementations has encountered trouble when teaming with groups who cannot speak English language conveniently (e.g.: Teams from Thailand, China etc.). Even if all members of a virtual team speak English, they may not speak the same English. For example, English spoken in the United Kingdom is different from United States. Pros and cons of common language are given below:

Pros: Easy to share thoughts in a common language

Cons: Accent-induced miscommunications

No Sea Language: Never speak 'Sea Language'; make sure that you are respecting majority people's cultural code and greetings, which is specific to geographic location. Try to avoid communicating in regional or local languages in client locations which looks peculiar to people. If an oral discussion or meeting involves several accents, it is always better to have written structure after the conversation. Bad spellings, use of abbreviations, heavy idiomatic phrases and usage of jargons may confuse people deviating them from understanding the exact content. Lack of fluency may lead to misinterpretations and misunderstandings. The west is more explicit and direct in their statements and so being straight and clear strengthens the conversations. In all conditions, it is good practice to summarize the meeting notes and publish across the intended circle.

Language Checklist for Distributed Test Team

- Be natural in your accent; don't try to imitate your client's accent
- Never speak in your regional language

- Don't do conversations in separate groups, especially in local languages when a con-call is going
- Avoid using jargons and idioms
- No abbreviations and local shot forms, though one of the distributed team member is from your region
- Use spell check before sending any docs/emails

2. Time zone Variances:

Temporal distance is a challenge when it comes to managing projects that are distributed length and width geographically. Any business with teams dispersed across the globe can literally 'work around the clock.' It's race against time to test managers to tackle the challenges of time zones while giving a large degree of freedom to their teams abroad. Pros and cons of time zones are given below:

Pros	Cons
Speeds up the projects	Leads to increasing waiting time for obtaining responses
Flexible scheduling	Lack of immediate support
Reduced calendar time	Time-zone-difference unavailability
Reduced Service level agreements and Turnaround times	Odd time Meetings
Fast solutions	Running daily meetings

C Language: The 'C' Language [Contact, Communicate, Co-ordinate and Calendaring] can storm out the odd differences in time zones. When you are good at 'C' language, you realize working across time zones is beneficial. While setting up your clocks for the day light saving, scheduling meetings through online calendars, following the standard times of various countries becomes a habit, Contacting, Communicating, Coordinating with team members becomes routine. Business quickly realizes the benefits of running a trade all round the clock.

Time Zone Checklist for Distributed Test Team

- Always note the time zones
- Keep watching out for daylight saving time
- Schedule meetings well in advance
- Communicate planned downtimes of applications

3. Geographical Conditions and Cultural Differences:

Due to location separation face-to-face interaction among the teams limits to electronic communication. Sharing of immediate information, seeking and providing instant support becomes complex for any ad hoc changes among dispersed team members. When the teams never meet, to maintain synchronous interaction, the team members should maintain discipline in handing off work to the team in a distant location, who picks up the work and continues it. Due to lack of visibility, status is derived mainly from what people contribute, not from title, affiliation or position.

Different national, organizational and professional cultural backgrounds have different work ethics, ideas, commitments and approaches. Bringing together divergent bodies of knowledge could lead to serious misunderstandings and conflicts. Every culture has its values and benefits and its own esteemed nature to respect them. Pros and Cons of geographical and cultural logical properties of a team are given below:

Pros	Cons
Shift works and working from home	Unavailability due to geographical weather conditions
Reduced travelling costs	Different holiday schedules and Religious background
Fewer relocations	Less face to face interaction
Opportunity for physically challenged	Assumptions increases due to vague information
24x7 work progress	Emotional conflicts
Diversified team with different cultures	Result in differences in values, perceptions, and work behaviors

See Language: Management (*onsite test manager and offshore test lead*) should always be alert, have control and knowledge of what's happening on the other world, which can be done through implementing 'See' language. Having a multicultural team is an extra challenge for every test manager. Test Managers need to keep track of resource availability irrespective of time zones and culture, especially for agile projects. If different languages are used, make an agreement on what common language will be used and never translate things more than once. Irrespective of type of project, resource region, location, culture etc., all the resources effort should be evaluated and recognized. Need to make sure the interaction of onsite and offshore resources communication is synchronous to business goals. Different religious backgrounds, work culture holiday schedules should not affect the team's performance. Maintaining advanced leave and holiday plans, finding replacements temporarily will not hinder the work progress.

Geo-location Checklist for Distributed Test Team

- Hand over emails, chats, texts
- Effective communication through meetings, videoconferences, teleconferences, webinars and recording
- Pay attention to silent members on your calls; know who contributes and who don't, who stays involved, and who stays in the background.
- Distribute information to all team members
- Common location for tracking day-to-day work

Culture Checklist for Distributed Test Team

- Respect all cultures
- Keep track of different holidays schedules and greet at appropriate situation
- Respect religious backgrounds

Tools

There's no shortage of advice on managing distributed teams, much of it focusing on the importance of communication and on the use of the array of available technical tools, such as WebEx, Skype, SharePoint, to aid in that process. Every distributed team must use some combination of these technologies to stay in contact and to make sure that the entire team is working from the same page. Teams that use these tools to communicate frequently and to overcome the obstacles of time and distance to stay on track and manage project progress and issues remove some of the risk from the distributed, or virtual, team model.

There are four essential tools for enabling real-time collaboration with a distributed test team:



- Instant Messaging
- Document Collaboration and version control tools
- Appointments managing tools
- Conferencing Tools

a) Instant Message Tools: Communicator from any service provider, like Microsoft, IBM, Yahoo, Google Talk, Skype etc.

b) Collaboration and Version Control Tools: White boards, sticky notes, Microsoft SharePoint (to manage tasks, bugs, documents), Microsoft Visual Studio Team System 2008/2010 (manage project templates, organize user stories, product backlog), Atlassian Jira (for agile projects), Google Docs, Jazz platform tools (like Rational Team Concert), RSS feeds (to publish frequently updated works), Wiki (to share information).

c) Appointment Managing Tools: FoxClocks plugin from Firefox, time zones sorting in Outlook, for scheduling use 'Time Trade' [timetrade.com], for converting/verifying time zones use 'The World clock' [timeanddate.com], Time Zone Converter [timezoneconverter.com], for reminders and managing appointments use 'Google Calendar' [google.com/calendar].

d) Conferencing Tools: Adobe Connect (conferencing and eLearning platform), Wikis (to share information amongst team members), CardMeeting (for distributed brainstorming), Windows SkyDrive (file sharing platform), Weave the People (customized, private and focused networks enable conversations to be centered around what is important to your team), Cross Loop (web conferencing and collaboration platform), BaseCamp (shared to-do lists, project plans and files), Skype and Babble ("Free" VOIP services), QNext (Free audio, video and document sharing), Convoq (Video, audio, screen sharing, presentation, IM, presence), last but not least 'WebEx'.

Do you know?

- DST - Most of the world does not take part in this practice. Much of South America, Asia and Australia do not participate in Daylight Saving Time anymore, while a large part of Africa never recognized it. Some U.S. states, including Arizona, don't even participate in DST. In some parts of Europe, DST is commonly referred to as "Summer Time." In Britain, it's known as "British Summer Time," while in Germany, it's "sommerzeit." Hawaii and most of Arizona don't observe the time change. U.S. territories that don't go on daylight saving time include American Samoa, Guam, Puerto Rico and the Virgin Islands.
- While many countries call public holiday, Britain's call it as bank holiday.
- Cross Culture is most chosen subject for research thesis presentation in most of the famous universities.

References:

Leading dispersed teams By Michael E. Kossler and Sonya Prestridge .CCL.

MST 512 – Open research papers on Project Management

The Driving Force: Lessons in Teamwork from Saturn and Other Leading Companies By Nancy Brown-Johnston, Xephor Press

Virtual team –Wikipedia - http://en.wikipedia.org/wiki/Virtual_team



Trinadh Kumar Bonam, MSc (Computer Science), Six Sigma (GB), CSM, CSPO, ITIL, HNC (from NIIT), has been in Analysis & Testing practice from past 8 years. His experience spans domains like Sales/Partner sales, Call

centre, Telecom(OSS/BSS/VAS), Retail, Manufacturing etc as an Test Analyst in various projects covering multiple technologies (like Siebel & MS CRM etc) and has been involved in Agile testing. He is now working for TATA Consultancy Services Ltd in US. His primary objective is 'QUALITY'. He is interested in learning new things, he finds never ending challenges in Testing.



Indrani Areti, alumni of Accenture, has 8 years of IT Experience. She served as a Release and Change Management Lead. Having started her career with Accenture, growing up to lead multiple projects with it, is an invaluable

experience for her. She played several roles and worked in various domains like Telecommunications, Banking, Finance, Volume Licensing, Retail sectors and more. Hailing from south east part of the world, working on the west side of the globe is a new experience for her now. She is passionate about writing specially in Software Testing domain.



Is That Testing?

- Katrina Edgar

When you are a tester who operates in a traditional environment it can be difficult to understand how testing could work any other way. Your organisation may use a tool that dictates aspects of your test process; test cases are written and linked back to requirements. Management may request test reports in a specific format, where number of test cases executed, percentage complete and bug counts are expected. There are often commercial drivers for testing being restricted to confirming that the software delivered meets what was requested. These are all barriers to changing how testing is perceived, not only by others in our industry but also testers themselves.

Testing is not simply verifying that requirements have been met. Testing is discovering and communicating information about software, which is difficult in the environment described above. Traditional testing represents a small portion of what testing is, so the confidence the business has in its outcomes is misplaced. As testers, we can educate ourselves and our organisations to deliver better testing.

The danger of confirmatory testing

Test cases that link to requirements focus the tester on proving that the application can deliver what it has been asked to do. This narrow scope can prevent the discovery of undesired behaviour in the application and revelations about what the business should have requested in the first place. Both these things are important in reducing the risk associated with releasing software.

A real user of the application will not know how it was requested to behave. It is the role of the tester to anticipate how customers will interact with the application and identify any problems they may encounter in doing so. Confirming requirements will not necessarily do this. To broaden your testing, you must ask questions beyond what is written in the requirements document. Conversations of this nature will reveal a number of new test activities.

Parallel analysis and execution

Increasing the scope of testing activities will mean that you need more time to test. Fortunately in a traditional process there is an obvious candidate for elimination. Writing detailed test cases is a wasteful activity, which often requires the tester to anticipate how the application will behave before interacting with it. When test analysis and execution are parallel activities, the tester has the opportunity to learn and alter their model of the software as they interact with it. If you stop writing pre-scripted test cases and instead identify only goals for your testing, you will create more time for real testing.

No test case does not mean abandoning evidence of testing. Instead of documenting what you plan to do, document what has actually been done. This can be achieved in a variety of ways and is a much more useful record for test auditing than a suite of test cases.

Similarly, no test case does not mean no structure. Rather, the structure of your testing can reflect the

needs of your project rather than the restrictions of a tool. Test analysis and critical thinking occur as before, but are now fostered to reach beyond the artificial boundaries dictated by test case templates and dependence on requirements. There are a number of heuristics to guide your thinking as you learn to explore with purpose; testing is not ad-hoc.

Real Reporting

Eliminating traditional test cases will mean that you can no longer count them, which will make many testers and project managers nervous. Test reporting has been synonymous with numbers for a long time. However when we interrogate what the numbers actually mean, we often find that they offer an illusion of testing rather than providing any real information.

The request for numbers stems from a desire to know how testing is progressing. Often a number is the most forthcoming piece of information from a traditional testing team, yet the very notion of reporting in this way should be as ridiculous as asking a business analyst "How many requirements do you plan to write and what percentage are already written?".

As the culture of testing shifts beyond verification of requirements, testers will have a greater understanding of the application. This is what management really want to know. Rather than telling them that testing is 62% complete, tell them a story about what you have discovered. Describe issues you have encountered and communicate potential risk. Be a source of rich, transparent and useful information. Reporting in this fashion will increase the value of testing as a service to the project team, which will alter the expectations of your colleagues about what testing is.

What is testing?

Testing in a traditional environment can change from verification of requirements to a broader scope. It can change from counting test cases to reporting useful information. Though the testing that you do should always be dependent on context, it should always be *testing*. As a profession, we need to share this common definition.

Testing is discovering and communicating information about software.

What are you doing?



Katrina Edgar is a tester from Wellington, New Zealand. She is a regular participant of KWST, co-founder and organiser of WeTest Workshops, and has recently been convinced to start blogging and

tweeting about testing. Find her online; @katrina_tester and <http://katrinatester.blogspot.co.nz>

Tips and Tricks for Testing Cloud Deployment Software



- Vivek Sharma

In the modern world of Information Technology (IT), Cloud computing, as a technological phenomenon, has generated a lot of buzz. All major IT giants have products that span across the Cloud Computing space. Amazon, in my opinion, has the largest portfolio of products that operate in the Cloud. Other vendors like Google, IBM, Microsoft, CA, VMware, and HP are not far behind. Additionally there are Cloud specialist vendors such as Salesforce, Rackspace etc. that have a good market presence too.

Various Cloud Computing vendors have products that span across facets such as Deployment (Installation) of a Cloud, Management of cloud, Support of cloud, Run applications on cloud and so on. It goes without saying that the domain of Cloud Computing is a complete ecosystem in itself.

This article focuses on the “Cloud Deployment” aspect of Cloud Computing. Specifically the emphasis is on some of the tips and tricks with respect to Testing of a Cloud Deployment Software application. A sample Cloud application would be ref-

erenced throughout to provide a more practical approach to the content that follows.

This article is *not* about software testing in the cloud; it is rather about testing of an application that is part of the overall Cloud infrastructure.

Terminology

- Cloud – Cloud means a pool of servers that present an aggregated set of resources (CPU, memory, storage and network) to host Virtual Machines. Cloud is a logical entity, while Cloud Server (defined next) is a physical entity
- Cloud Server – Cloud Server is a physical server that is located in the datacenter. The Cloud server is a server class machine that has the capability to support virtualization
- Cloud Deployment Software (CDS) – Cloud Deployment Software is a software application that installs the Cloud and the Cloud servers that are part of it. Throughout the course of this

article, Cloud Deployment Software would be abbreviated as *CDS*.

- Datacenter- A datacenter is a facility used to house computer systems and associated components, such as telecommunications and storage systems. It generally includes redundant or backup power supplies, redundant data communications connections, environmental controls (e.g., air conditioning, fire suppression) and security devices .
- Virtualization - Virtualization is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system (OS), storage device, or network resources.
- Server Virtualization - Server virtualization is a type of virtualization that involves abstraction of server resources from server users. The server administrator uses a software application (hypervisor) to divide one physical server into multiple isolated virtual environments often called as Virtual Machines (VMs).
- Hypervisor – Hypervisor is the software application used to divide one physical server into multiple isolated virtual environments (VMs). Examples of hypervisors include VMware ESX Server, Microsoft Hyper-V etc.
- DRAC – An acronym for Dell Remote Access Controller. DRAC is an interface card from Dell Inc. that provides out-of-band management facilities. Key features include power management, virtual media access and remote console capabilities, all available through a supported web browser or command line interface.
- PXE - Preboot eXecution Environment (PXE, also known as Pre-Execution Environment; sometimes pronounced "pixie") is an environment to boot computers using a network interface independently of data storage devices (like hard disks) or installed operating systems.

The above definitions have been taken from the following sources:

- en.wikipedia.org
- searchservervirtualization.techtarget.com

Note: If you are new to Cloud Computing, it is recommended to read NIST definition of Cloud Computing, presently available at [this](#) link.

Sample Application

This section provides details of a (sample) Cloud Deployment application – starting from its features, basic deployment architecture and theory of operation.

Features of Cloud Deployment Software (CDS) application

- Provides a single point of configuration for the Cloud which includes – Installation* (deployment), Upgrade, Patching, Scale up/down the Cloud.
- Provides a web user interface for the administrator
- Monitors the health of Cloud through reports such as hardware utilization, Cloud server status, license status etc.
- Manages the assignment of public and private IPs to Cloud servers
- Provides a set of APIs to install the Cloud
- Provides options to perform power management operations on Cloud servers, for any maintenance
- Provide recovery capabilities for CDS (from its database replica), in the event of a disaster

* => The Cloud once installed can be used by the Cloud Service Provider as a Public or Private Cloud. However for the sake of simplicity of this article, let us assume that the Cloud being installed would be used as a Private Cloud.

Deployment architecture

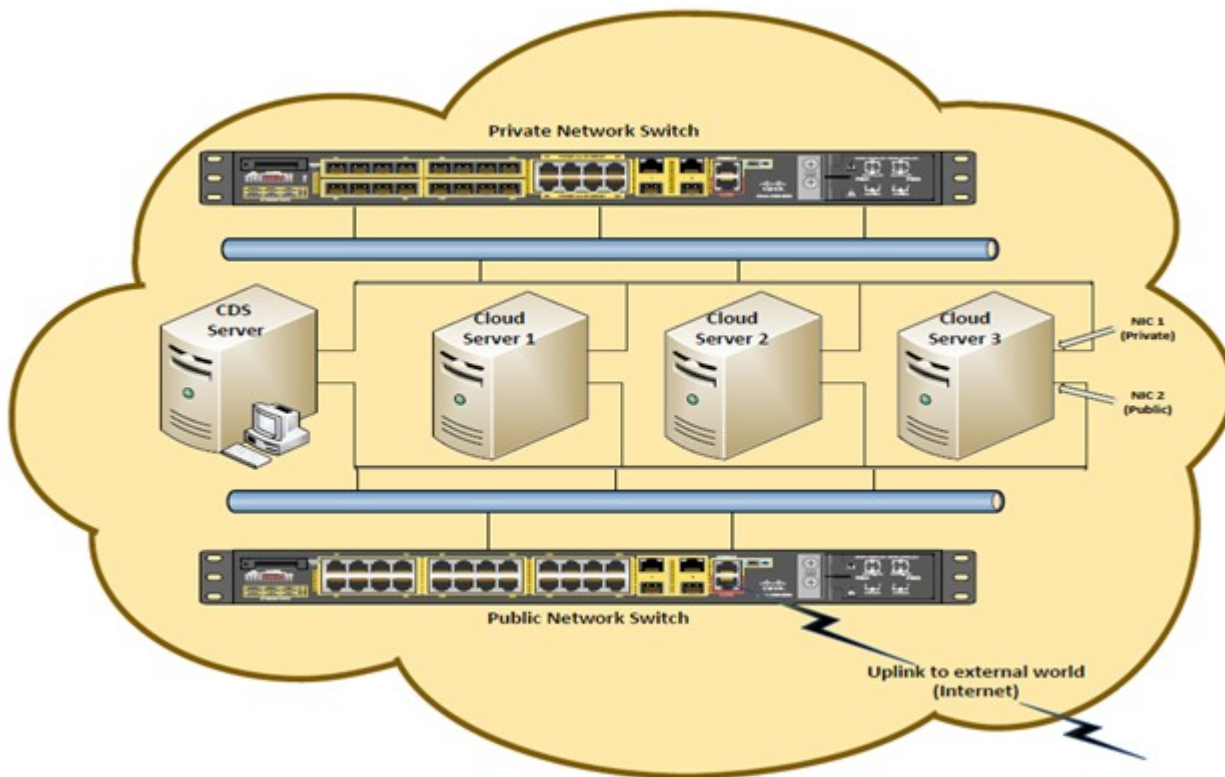


Figure 1 - Layout of Cloud Deployment Software along with the underlying infrastructure

- **CDS Server** – This server runs the CDS application. The server has two Network cards – one connects to the private switch and other to public switch
- **Cloud Server 1 to 3** – These three servers will run the Cloud application that includes an operating system, a hypervisor and Cloud vendor specific packages. The Cloud servers have two **Network Interface Cards (NIC)** – one connects to the private switch (NIC 1) and the other to the public switch (NIC 2). Each server has an additional DRAC power card for remote power management operations like power off, power cycle etc.- that connects to another switch (not shown in the diagram)
- **Private Network Switch** – This is a network switch that is dedicated for communication among the Cloud servers as well as between CDS and Cloud Servers.
- **Public Network Switch** – This is a network switch for any user who requires access to an application that is hosted on the Cloud Infrastructure. For example – assume gmail.com is hosted as an application (SaaS) on this Cloud infrastructure; any users who need access to their gmail account would connect via the public switch.

The public switch has an uplink to internet that provides connectivity to the external world

Theory of operation

CDS is a web application, built on a Linux based solution stack and comprises the following:

- Linux – Operating system
- Web server (like Apache)

- Database (like MYSQL)
- User interface accessible from a browser

followed by hypervisor installation and finally Cloud application vendor specific package(s) installation

With respect to the above mentioned deployment architecture, the main steps in order to create a fully functional Cloud are as follows:

- Install CDS application on the designated CDS server, which involves:

Configure the installation prerequisites on Linux machine (CDS Server) → Install CDS

- Install Cloud packages (operating system, hypervisor, application packages) on the designated Cloud Servers, which involves:

PXE boot Cloud servers over private network → From CDS web UI/API, submit Cloud deployment job → CDS performs network based operating system installation on cloud servers,

The Testing Tips

This section is the nucleus of the article. Having learnt about the sample CDS application, it is time to cover the Testing Aspects of such application. Following are the various Types of Tests applicable to CDS:

- Installation
- Functionality
- User Interface
- Performance
- Security

Here are a few questions for each Test Type that would help to determine the Test Scope

Test Type	Question(s)
Installation	<p>What are the supported Linux distributions for CDS installation?</p> <p>What are the installation prerequisites?</p> <p>What are the CDS upgrade paths?</p>
Functionality	<p>What is the flow of Cloud installation through CDS?</p> <p>How does CDS communicate with the hypervisor on Cloud Server?</p> <p>Does CDS allow addition and removal of Cloud servers (scale up and down) on the fly?</p> <p>What are the monitoring capabilities of CDS?</p> <p>How does CDS handle error scenarios like a Cloud installation fails due to power failure on one of the servers?</p> <p>Does the installed Cloud continue to operate in case CDS is unavailable?</p> <p>What are the various CDS APIs to install the Cloud?</p>
User Interface (UI)	<p>What are the supported browsers and devices for CDS UI? Is the UI localized?</p>
Performance	<p>What is the maximum number of Cloud servers supported for a single Cloud?</p> <p>What is the maximum number of Clouds that a single CDS installation can deploy?</p> <p>What is the time taken to create a Cloud with <n> servers?</p> <p>What are the benchmarks for UI performance?</p> <p>How does CDS behave when subjected to certain load?</p>
Security	<p>What are the security aspects of CDS that should be taken into consideration while testing?</p>

An answer to each of the above questions would yield two outcomes:

- a. It would help to decide the Testing Scope of CDS
- b. It would lead to more questions that would help to further gain an understanding of CDS Test Scope

Determining the Testing Scope is the most important factor to devise an effective Test Strategy. Once the scope and strategy are finalized, it is time to focus on the actual test execution. The table mentioned in the next section talks about my experience within the Test Execution phase.

The next section is based on my experience within the Test Execution phase of an application similar to the CDS sample application. The information is represented in the following format:

Issue 1

Title: CDS installation fails with invalid message due to an erroneously configured prerequisite

Description: One of the prerequisites for CDS installation is a properly configured `/etc/hosts` file on the Linux machine. For this issue, the `/etc/hosts` file had an entry in below format

```
<hostname> <IP_Address>
```

CDS installer was unable to retrieve the hostname of the system (`hostname -s` returned an unexpected output) and failed the installation with an invalid message

“Unknown error”

Testing Tip: Ensure that you include error handling scenarios in the installation tests. There are cases where a mis-configuration may not be handled by the installer and it behaves abruptly.

Issue 2

Title: ssh login to CDS server is blocked after CDS is upgraded

Description: The issue happened during upgrade of CDS to a newer version where the installer cleared an entry in

`/etc/ssh/sshd_config` file that prevented any new ssh login to the CDS node. The issue surfaced because of a new implementation in the new version where an extra check was added in the installer to verify that password based login is disabled for CDS node and user can only do ssh key based login. For a fresh installation, things were fine; however an upgrade path caused an issue.

Testing Tip: Always keep a note of any changes/enhancements made to existing functionality in a newer version of a product and understand its impact for the current users.

Issue 3

Title: DRAC* login for Cloud Server was blocked after it started getting managed by CDS

Description: The issue surfaced due to a hard-coded value. As soon as a Cloud Server is under management control, CDS creates a power management user for that Cloud Server in its DRAC Users list. The issue was that the CDS specific user was always created in slot #2 without checking whether that slot is already in use or not. As a result, if slot#2 on Cloud Server's DRAC user configuration initially had a user that was used for DRAC login (like root or another account); the same user no longer existed after the Cloud Server was managed by CDS – hence login got blocked

*For details about DRAC, refer Terminology section.

Testing Tip: Understand the integration of your product functionality with third party tools/technologies. . In this case – CDS has integration with Dell's DRAC technology and an issue in CDS caused an error in operation for DRAC

Issue 4

Title: Cloud installation fails on particular hardware due to missing drivers

Description: This kind of issue is expected to surface across Cloud products that are supposed to run on a variety of hardware. In this particular instance, for a particular hardware model, the drivers were not bundled along with the Cloud installer operating system image, as a result of which the Cloud installation failed

Testing Tip: Prepare your Test Matrix in accordance with the kind of production environments where your Cloud

is going to run (refer Appendix A). Keep yourself up-to-date with the latest in the field of computer hardware – through sources such as hardwaresecrets.com, tomshardware.com

Issue 5

Title: CDS caused data loss on a Cloud server, as part of recovering it from a power failure

Description: This issue represents an edge case that led to data loss. The issue was that a functional Cloud server had a power failure and later as part of recovering the Cloud server from the failure, CDS started reimaging it with a new operating system thereby erasing all disk partitions and the data

Testing Tip: Ensure to add error handling scenarios as part of your overall Test Strategy. Such scenarios may not happen on daily basis, but could be catastrophic when they occur

Issue 6

Title: Upgrade failed when an existing Cloud installation had large number of servers

Description: This issue happened in a Cloud setup containing 20 Cloud servers at v1.0 being managed by CDS. When a command was issued by CDS to upgrade the Cloud to v2.0 – the whole upgrade operation failed. The issue was due to the way the upgrade process was designed. The upgrade happened in batches of eight Cloud servers at a time à after the first eight were upgraded they were rebooted à while they were getting rebooted the remaining 12 servers lost connection with the primary Cloud server (since it was present in the first batch of 8 servers) à As a result, all the 12 servers rebooted themselves causing the entire Cloud upgrade to fail

Testing Tip: Ensure to test upgrade paths in large environments that simulate production environments. In the above issue, a Cloud upgrade would succeed if the Cloud setup consists of <= 8 servers; however it failed when tested with a larger count

Issue 7

Title: Log file overwritten upon CDS service restart

Description: This issue happened where CDS was operational for a few days and multiple copies of the log files had been generated in /var/log directory.

Example - CDSservice1.log and CDSservice2.log files were already generated and filled to their maximum capacity (10 MB in this case) and CDSservice3.log was the currently active log.

In such a scenario, if the CDS service was restarted, the logging stopped into CDSservice3.log and started writing into CDSservice2.log. As a result the older log info in CDSservice2.log was overwritten and hence the issue.

Testing Tip: Ensure to include longevity tests as part of your test strategy. As part of those tests, verifying entries in log files such as timestamps, content, log file rotation etc. should be covered

Issue 8

Title: API call hangs due to improper cleanup of previous API call

Description: This issue happened as an API call was made initially with an invalid parameter. That call got stuck in a continuous loop, thereby blocking any further API calls. The fix was to handle the bad API call to terminate itself immediately with an appropriate error message, so that subsequent calls can run fine.

Testing Tip: Ensure to include error handling scenarios as part of your API testing. A mix of valid and invalid calls can sometimes confuse the applications and it starts returning incorrect return codes

Issue 9

Title: After CDS upgrade, UI screens in localized environment (French) still show up in English

Description: This issue surfaced as one of the resource file was not being updated correctly after CDS upgrade. As a result, when the CDS UI was viewed in a French locale – the text in some of the screens showed up in English.

Testing Tip: Identify the localization testing needs of your product and ensure they are part of your overall Test Strategy. The focus should be to verify that the strings are properly translated into the respective language(s) for fresh installation and upgrades

Issue 10

Title: Cloud status wrongly shown in UI, in special cases

Description: This issue was a result of trying an operation that led to a mis-communication between CDS UI and its back-end. The steps were as follows:

Submit a Cloud creation from CDS UI → While the Cloud status is “Creating Cloud”, submit Stop Cloud command (usually applicable when Cloud status is “Running Cloud”).

This resulted in a mismatch between the UI component that gathers Cloud state and the back-end component which still thinks that Cloud creation is in progress. As a result Cloud state in UI continued to report as “stopping Cloud creation” indefinitely.

Testing Tip: While routine functional tests are usually part of the testplan, it is good to try out non-common paths to verify product behavior. In the above example – attempting a Stop Cloud operation is usually done for a Running Cloud; however for a Cloud that was still getting created, attempting the same operation led to the issue

Issue 11

Title: Cloud server discovery failed when CDS public switch port LAN differs from that of Cloud Server

Description: This issue is related to the network switch configuration wherein CDS tried to reach a Cloud server over public network and was unable to do so, since the Cloud Server switch port was configured with a different VLAN. The issue here was that CDS tried to communicate with a Cloud server over the public network, which violates the underlying design wherein all communication between CDS and Cloud Servers should occur over a dedicated private network

Testing Tip: Understand the domain around which your product operates and gather knowledge on it. That could be handy, for instance, the issue described can be identified only if the tester has knowledge of Networking domain.

Issue 12

Title: CDS hangs when maximum supported Cloud installations are kicked off

Description: This issue caused a deadlock situation where CDS service hung as it reached maximum thread limit and it ultimately caused the Cloud installation jobs

to hang. The issue happened when multiple Cloud installation jobs were submitted through CDS API

Testing Tip: Ensure to test the dimension limits of your product. I have seen instances where such tests uncover issues that could lead to a design changes in the product.

Issue 13

Title: CDS UI shows incorrect Cloud server state in a large Cloud setup

Description: This issue happened due to timeout issues while CDS was trying to gather Cloud server state for a 40 server Cloud. For CDS: The timeout value to gather state of all servers in a single Cloud was 120s. In this case the time taken to gather the state of 40 Cloud servers exceeded 120s – hence the issue.

Testing Tip: Timeouts are a common cause of issues in several different products. It is good to gather information about the timeout values for various operations within your product, as applicable and plan your tests accordingly.

Conclusion

The advent of Cloud Computing has brought a paradigm shift in technology. The good news is that for enthusiastic Test Engineers, this brings spectacular challenges in their day to day job. From testing perspective, one needs to consider multiple factors for testing a Cloud based application from performance, scalability, hardware infrastructure (esp. networking and storage), virtualization, and so on.

Cloud Deployment Software (CDS) is one of the many components in the big Cloudy world. This article is an attempt to put forth my experiences with the testing of Cloud Deployment Software and the lessons learnt out of it. To conclude, here is an analogy from an ancient story (which was told by my honorable mentor Siva Palagummi) in The Mahabharata:

Arujuna's son Babruvahana learnt few innovative techniques in archery from Satyabhama (Lord Krishna's wife). During some altercation he ends up in a fight with Lord Krishna himself. In that battle Krishna fires Nag astra (snake weapon). Usually the counter for Nag astra is Garudastra (eagle weapon). Had Babruvahana fired Garudastra, Krishna could have countered it with Vaishnav Astra (another potent weapon).

Instead, Babruvahana smartly fires Pipilika astra (weapon of billion ants) for which counter is not known and wins the battle

Analogy

- **Nag astra** in the story resembles the problem that we are trying to solve through “some” technology
- **Garuda astra** (Eagle weapon) resembles the Mainframe – large scale computer system, that can single handedly perform bulk data processing.
- **Pipilika astra** (weapon of billion ants) resembles the Cloud – which consists of a pool of servers that present an aggregated set of resources to perform a task. Even if one server goes down, there is another one to take over (Highly Available) and continue the data processing.

Appendix A – Cloud Deployment Software Test Matrix

The test matrix for CDS consists of two parts:

- CDS Installation

- Cloud Installation

CDS Installation Test Matrix

The CDS installation matrix is created based on the following input parameters:

1. Install Mode – Interactive, Silent
2. Linux Distro – Cent OS, Ubuntu, Suse Linux
3. Hardware Model* – Dell PowerEdge, HP Proliant, SuperMicro, IBM System x, Fujitsu
4. CDS Database Replication – Enabled, Disabled

If we consider all combinations, there would be a total of 60 tests. Following table depicts a risk-based approach for the tests that would cover the various combinations:

Test Number	Install Mode	Linux Distro	Hardware Model*	CDS DB replication
1	Interactive	Cent OS	Dell PowerEdge	Enable
2	Silent	Ubuntu	Dell PowerEdge	Disable
3	Silent	Suse Linux	Dell PowerEdge	Enable
4	Interactive	Cent OS	HP Proliant	Disable
5	Interactive	Ubuntu	HP Proliant	Enable
6	Interactive	Suse Linux	HP Proliant	Disable
7	Silent	Cent OS	SuperMicro	Enable
8	Interactive	Ubuntu	SuperMicro	Disable
9	Interactive	Suse Linux	SuperMicro	Disable
10	Interactive	Cent OS	IBM System x	Enable
11	Silent	Ubuntu	IBM System x	Disable
12	Interactive	Suse Linux	IBM System x	Enable
13	Interactive	Cent OS	Fujitsu	Enable
14	Silent	Ubuntu	Fujitsu	Disable
15	Silent	Suse Linux	Fujitsu	Disable
16	Silent	Cent OS	HP Proliant	Enable

* => The reason to consider this as a matrix is for case where CDS itself is bundled as an application along with a thin operating system (like BusyBox) that gets installed on Bare Metal hardware.

Cloud Installation Test Matrix

The Cloud installation matrix is created based on the following input parameters:

1. Server Model – Dell PowerEdge, HP Proliant, SuperMicro, IBM System x, Others..
2. Installation Method – From CDS UI, From CDS API
3. Hypervisor – Xen, VMware ESX
4. Storage – Local, NFS
5. NIC model – Broadcom, Intel, Cisco
6. NIC speed – 1G, 10G

If we consider all combinations, there would be a total of 240 tests

Following table depicts a risk-based approach for the tests that would cover the various combinations:

Test Number	Server Model	Installation Method	Hyper-visor	Storage	NIC model	Network Speed
1	Dell PowerEdge	From CDS UI	Xen	Local	Broadcom	1G
2	Dell PowerEdge	From CDS API	ESX	NFS	Intel	10G
3	Dell PowerEdge	From CDS UI	ESX	Local	Cisco	10G
4	HP Proliant	From CDS API	Xen	NFS	Broadcom	1G
5	HP Proliant	From CDS UI	Xen	Local	Intel	10G
6	HP Proliant	From CDS UI	Xen	NFS	Cisco	1G
7	SuperMicro	From CDS API	ESX	Local	Broadcom	1G
8	SuperMicro	From CDS UI	Xen	NFS	Intel	1G
9	SuperMicro	From CDS API	Xen	Local	Cisco	10G
10	IBM System x	From CDS UI	Xen	Local	Broadcom	10G
11	IBM System x	From CDS API	ESX	NFS	Intel	1G
12	IBM System x	From CDS UI	Xen	Local	Cisco	1G
13	Others	From CDS UI	Xen	Local	Broadcom	1G
14	Others	From CDS API	ESX	NFS	Intel	10G
15	Others	From CDS API	ESX	Local	Cisco	10G
16	HP Proliant	From CDS API	ESX	Local	Broadcom	1G

Notes:

- a. The tables have been prepared through tools available at hexawise.com
- b. The above table is just a reference, but may not reflect the compatibility of a server model with the corresponding NIC model. Refer vendor specific website for any compatibility checks.



Vivek Sharma works with CA Technologies as a QA professional. He has over 11 years of experience and is inclined towards testing of products based on Virtualization and Cloud related technologies. Besides software testing, he is fond of tennis and football. Vivek lives in Hyderabad along with his very supportive wife and charming daughter. The opinions and statements in this article are his own and do not necessarily reflect the opinions or policies of CA and are based on his experiences with a variety of software products and companies.

41 Definitions of Software Testing

- Jyothi Rangaiah

What is Software Testing? – Heck! Define it yourself.

I have been thinking about an answer for the same but have found it hard to convey in one statement.

Am I a born, reborn or a resurrected tester?

I hope to find out for myself and the below article is an attempt at this.

Testing - I will be using the word testing as I continue to write and as you readers read, read it as Software Testing.

1) Testing is a responsibility of representing information which is an essential necessity for bettering the application/product under test.

2) Testing is to learning to think well.

3) Testing is to understand the various contexts a system can be applicable in.

4) Testing is identifying the subtleties and extremities where the system can be used.

5) Testing is craving to dig deep into the system to look in the nook and corner to project the information that can awaken the product owner and the user to surprises and a wow-ness that a product can be used to perform.

6) Testing is to provide the consumer with an application which re-ensures confidence in the consumer and for the business.

7) Testing is that ability which the whole team is entitled to with an opportunity to grab the consumers attention, supply the consumers demand and to deliver well.

8) Testing is to convert that dormant thought into an active on-going action oriented process.

9) Testing is to continuously collect aids which aims at delivering quality information to anyone equipped to better build the product.

10) Testing is remembering to act in unison with the vision and mission reflecting in the consumable product.

11) Testing is questioning, challenging, being biased and up-rooting the biases about how the product is presumed to be built and used.

12) Testing is having an eye for details however miniature or magnanimous.

13) Testing is buying yourself a microscope and a telescope to look at how a product is consumed today and in future.

14) Testing is building a lifelong insanity to learn in all sanity.

15) Testing is a role play of that of an investigator, a doctor, a builder, a victim, a crime fighter, an intruder, a seeker, an evangelist, a doer.

Do you see such attributes in a tester? - Hire that person.

16) Testing is being in a context all assuming and continuously judging.

17) Testing is testing the assumptions and then falling prey for the judgments made.

18) Testing is re-opening a concluded case.

19) Testing is to don the hat of someone other than you, change perspective and test with a prejudice.

20) Test to KNOW.

21) Testing is time boxed and at-times unleashing the you, learning to think in a way which is not brand you.

22) Testing is building credibility for yourself, your organization which serves you and which you are serving.

23) Testing is learning to explore the path which you are willing to tread and paths which are road less travelled.

24) Testing is defining, redefining and un-defining.

25) Testing is breaking barriers to test.

26) Testing is a courageous act of preparing oneself to tread a new path, take another challenge.

27) Testing is taking ownership of mistakes with a pitcher of gratitude, that I learnt what not to do and what to do in this context.

28) Testing is story telling via testing and the experience reports.

29) Testing is diminishing confusion and expanding the confidence of a user.

30) Testing is that walk down the memory lane and think if this issue has occurred or is a déjà vu.

31) Testing is that feeling when you love yourself for learning to learn new every day.

32) Testing is together untying and revealing the product/application to itself.

33) Testing is you emerging out of the bath tub with a 'Eureka' moment.

34) Testing is a knock on the door of a developer to help undertake measures to provide a fix.

35) Testing is a wakeup call to innovation, to time travel back into the future.

Did you relate to any one or more of these?

36) Testing is an unconventional mode of transport to the minds of a user.

37) Testing is at-times masking the status quo.

38) Testing is closing in on the 'I' the consumer, 'I' the tester, 'I' the developer, 'I' the owner and illuminating the path of 'We' the team.

39) Testing is that run down the rabbit hole to discover the wonderland of Alice/Alfred to sketch the tomorrow of testing.

40) Testing is Learning.

41) Testing is Circus. (*This one is sponsored by Testing Circus team.)

Am sure you have moulded yourself into a tester with your own definition of testing and tester, do share your thoughts on the same.

Come, join and be a part of this community of information seekers and providers.



Jyothi Rangaiah, a trespasser into the minds of users, dons many hats that a tester should in order to test.

She who has made learning a way of living. Continues to inspire herself by constantly connecting with the learned from the newly introduced community of

testers.

Willing to make way, where there is no way she finds herself fighting the crime scene in her vicinity. And on this journey bugs bump into her.

Unwilled to take no for an answer without judgement, reasoning and questioning has helped her in her way of testing. Jyothi Rangaiah is a budding blogger who writes at chroniclesoftesting.blogspot.com. A fan of defining art who's favorite English word is genuine. Follow her whom challenges constantly follow on twitter @aarjay

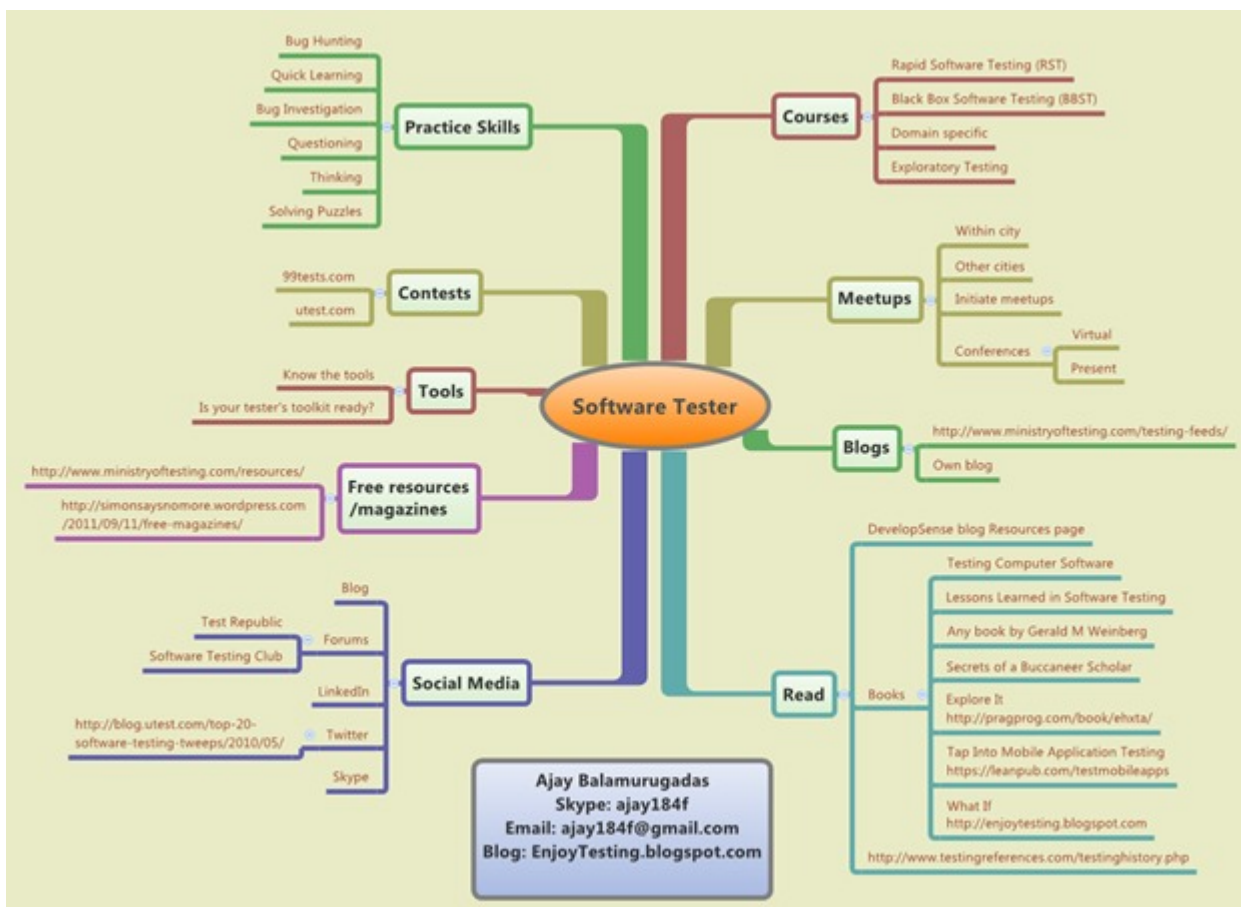
The “CMB RSM TCP” Software Tester

- Ajay Balamurugadas

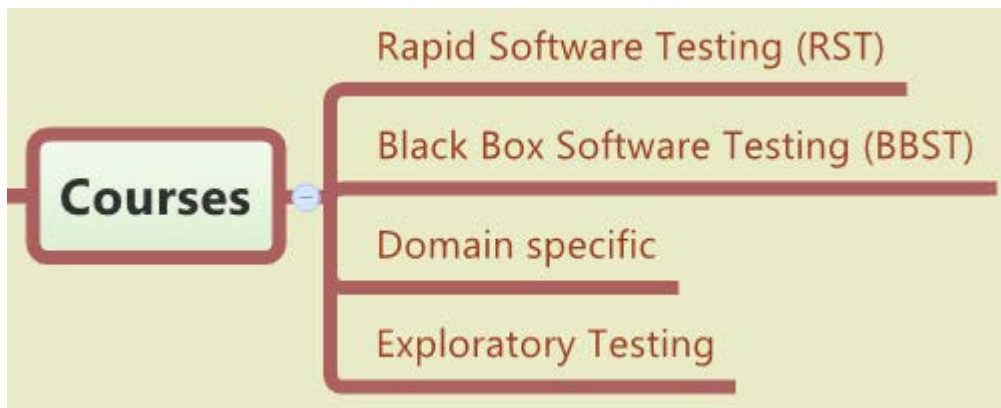
As I write this article, I have spent seven official years testing software. Do I know to test software? Am I a good software tester? I still don't know the answer. I am still learning and continue to practice software testing. So, what is the article about? Through this article I attempt to highlight the key points that will help a software tester improve his/her skills. And once again mind maps come to my rescue.

There is definitely no shortcut or defined process to achieve excellence in testing skills. And as **Jerry Weinberg** quotes here:

<http://blog.ute.com/introducing-the-top-secrets-of-top-testers/2013/06/>



“Each person is different, so spend your time developing your own skills as a tester.”



Here is a note for you readers. Make sure to know in detail about any word highlighted in **bold** in this article. I believe in continuous learning. One could learn about any topic on their own or learn from someone who is experienced about the particular topic. There are so many courses available today that testers should consider themselves lucky

to get so many opportunities to improve themselves. Some of the ones which have had an impact on me are as follows:

Black Box Software Testing (BBST):

I learned about finding bugs, investigating bugs and understood mission, context, **bug advocacy** much better after completing the BBST courses. The course content is available here: <http://testingeducation.org/BBST/> . I feel that every software tester should complete BBST courses.

Rapid Software Testing (RST):

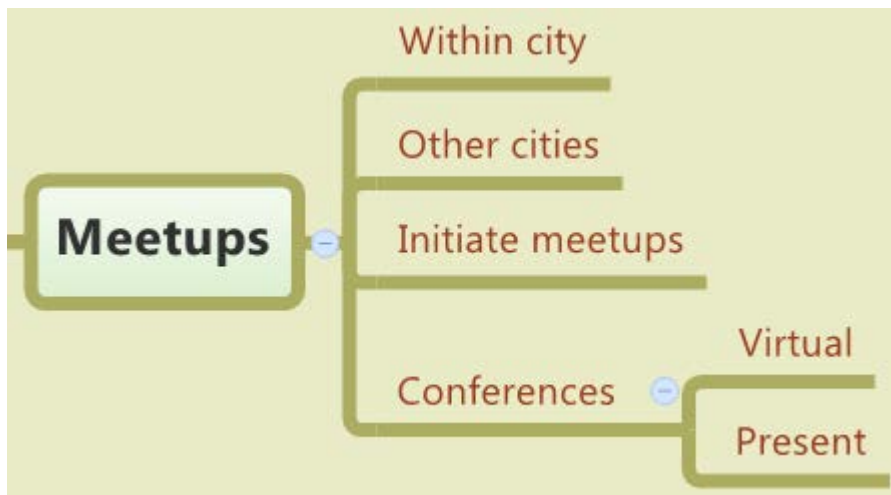
The first course I attended as a software tester. This taught me about **heuristics**, **mnemonics**, **oracles** and testing skills. If you have not read the RST course slides (<http://satisfice.com/rst.pdf>) yet, you are missing a lot.

Domain Specific courses:

There are other courses which focus on a specific domain - say web security or mobile testing. Based on your needs, attend these courses.

Exploratory Testing:

This course is about an approach - Exploratory Testing which can be applied to any testing technique. I and Dr. Meeta Prakash conducted a workshop on Exploratory Testing in World Conference on Next Generation Testing at Bangalore.



Attend meetups conducted in your city. If you cannot find one, initiate one. There are monthly meets conducted in Chennai regularly. Search for events. For ex: <http://events.doattend.com/city?name=bangalore> will give me results for meetups in Bangalore. Also, there are virtual conferences like <http://www.eurostarconferences.com/content/free-virtual-software-testing-conference> Practice your **communication skills**, submit your abstracts to conferences and present in a conference. If you need a lot of practice,

start with local meetups, **lightening talks**, office meetings and then move on to software testing conferences.

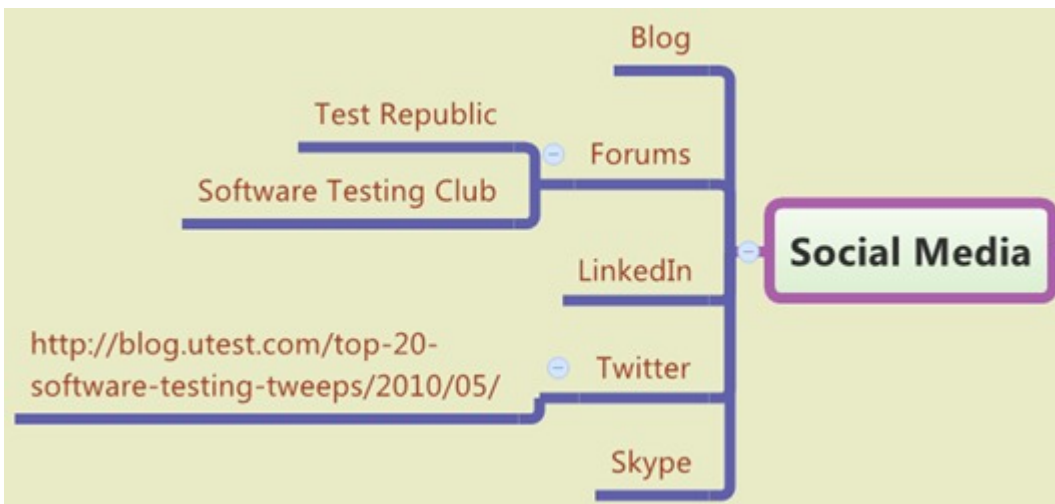


A good software tester reads and interacts with lot of other testers. One of the easier ways to know

about others' thought process is to read articles by them. A blog is a very good start. Which blogs do you read? If you are complaining of the number of blogs to keep track of, I hear you. Here is my suggestion: Every day, watch out for the updates on the testing feeds by **Ministry of Testing**. This would not take more than thirty minutes per day. You should start your own blog too. Write about your experiences and one fine day you will have a reader base which will help you improve your writing and thinking skills.



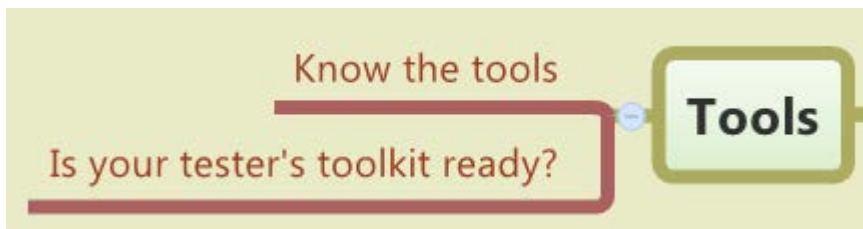
Read the **books** mentioned in the image. **Michael Bolton's blog** resources page: <http://www.developsense.com/resources.html> has a lot of useful information. Many more resources are linked from this page. Go through each of those articles. Do not get overwhelmed and take one article at a time. It is a good idea to be aware of **testing history** too.



It is a good practice to have the same id across social media. Grow your network. You never know who will share the common interest or who will get your next business deal. When your name is Googled, do you see hits related to your testing career or everything other than testing?

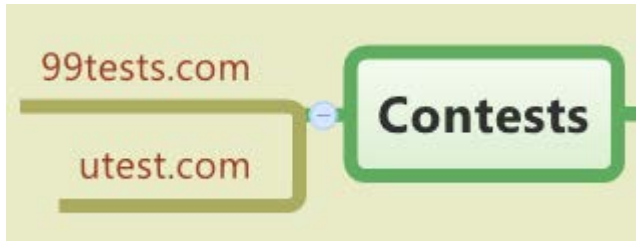


Isn't it wonderful if you get valuable resources without paying any money? The two links are examples of that. Feast on the articles.

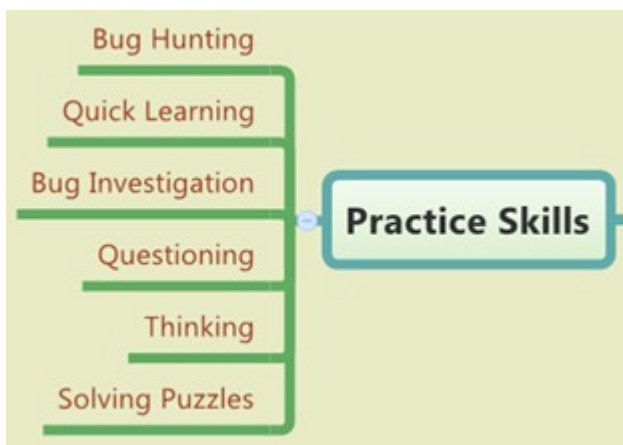


How many tools are you aware of? Can you use them effectively right now? When was the last time you used the tool and do you know which tool to use in which situation? Build your own **tester's toolkit**. Scout for tools. Have a pen drive ready with tools, **cheat**

sheets, test files and other articles that can help you in testing.



I have met good testers and have learned new test ideas by participating in **99tests** contests. I have also heard good things about **utest**. Participate in the contests conducted by both of them. There are other contests like **Zappers** too.



Finally, keep practicing these skills. There are many more skills to be added but let us be good at these to start with. **Weekend Testing** (www.weekendtesting.com) is a good platform to practice testing skills.

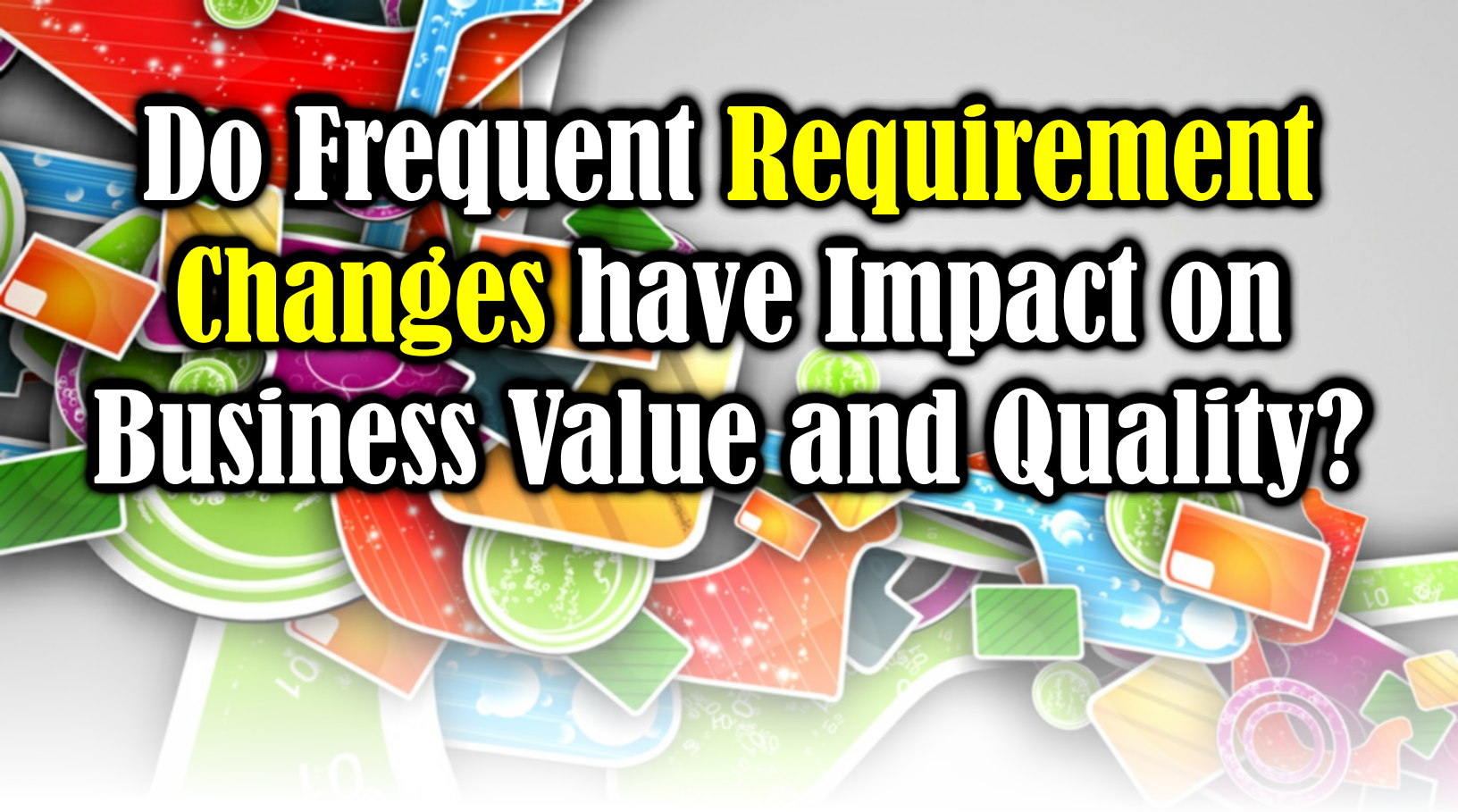
With regular and right practice, I am sure that your testing skills will improve a lot. My best wishes to each one of you. Till I meet you with my next article (writing skills), let me practice.



Ajay Balamurugadas, a software tester loves software testing, interacting with programmers and product management to help deliver a great product to the market. He understands that he is offering a service and is not the gatekeeper of quality. His thought process can be read through his blog posts and short books. Feel free to call him to your city for a workshop or a talk to help your testing team.

Skype/Twitter: ajay184f

Blog: www.EnjoyTesting.blogspot.com



Do Frequent **Requirement Changes** have Impact on Business Value and Quality?

- Rrajesh Barde

Change refers to the process of becoming different. Change brings in more changes on multiple aspects and one such aspect is Requirements change. Software Development Life Cycle undergoes many changes through each of its phase. Each phase of SDLC has many areas and each such area undergoes change often. Some changes are valid and beyond control, geared to meet the business requirements but some changes are detrimental to the development effort. When it comes to testing, the harmful effects of frequent changes are more damaging.

Value addition to a testing initiative is more when there are less or no changes to requirements, to the application under test or the test artifacts such as test plans, test scenarios, test cases etc. Frequent changes result in frequent updating and tracking. Change that happens frequently affects the test life cycle directly. This causes a ripple effect on test analysis and planning, approach and strategy, scenario identification and test design, test case preparation, test execution and reporting. Finally, quality suffers at the hands of frequent requirement change, be it valid or invalid from the business perspective. What matters here is the business value that comes from quality. And to maintain that quality, one has to be focused on gathering relevant business and software requirements and freezing them before it is released to design, coding and testing.

Changes to requirements are imminent when a project starts. There are bound to be challenges in handling those changes. But the real business value depends on which phase of SDLC do the changes come in, what are the severities of the changes, what is the impact on the schedule, resources, costs and the ROI.

This paper aims to uncover some myths and facts about business value in testing through frequent requirement changes and its impact on quality.

INTRODUCTION

Fred Brooks, father of OS/360 was quoted as saying that “changeability is one of the essential difficulties of software production.”

In IT terminology, the word “change” means an advantage to the customer and a challenge to a software developer. Customer sees change in requirements as a business need but a software developer sees it as a risk. When requirements change is a risk for a software developer, then it is a bigger risk and a bigger challenge for a tester. From a testing perspective, any change to requirements, be it business or software requirements, is a risk unless it is managed well. Since, testing is a phase closer to production; last minute change in requirements deals a huge blow to the quality of the product. It is well understood in software life cycle that a requirement change is synonymous with the development and that it is a need from a customer who pays for the product.

When a customer, who is the sole owner of a product, requires changes in the way the product should fulfil its business needs, then the required changes have to be made.

Then what happens to the Business Value when those necessary changes have been made? Do those changes directly impact the quality of the product?

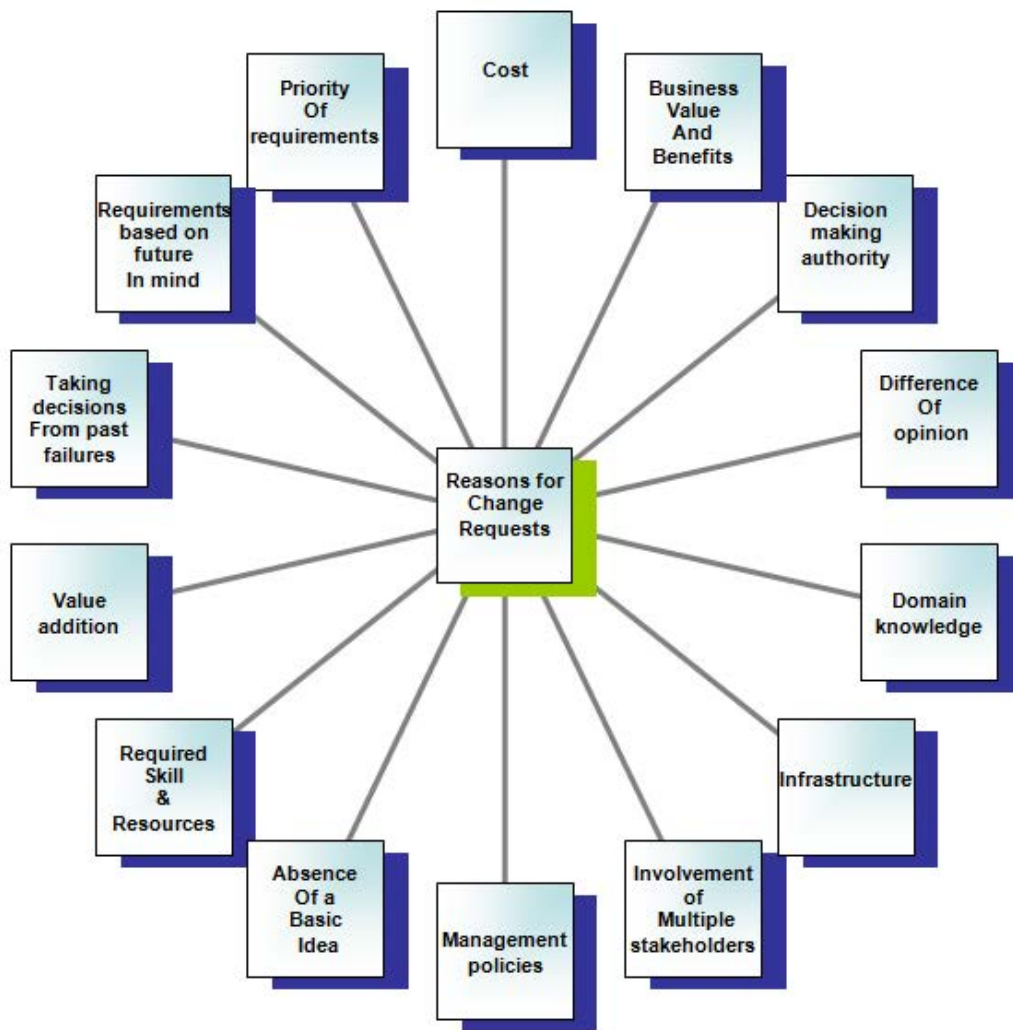
REASONS FOR REQUIREMENT CHANGES

Requirements can change for various reasons and at any time. It is part of SDLC that changes happen to requirements. Change takes place when the very basic “idea” of software development is sometimes questioned. It goes to the extent of “who did this” and “why this is like this and why not like that”.

Some valid reasons for changes in requirements could be:

- Business value and benefits
- Cost
- Decision making authority
- Difference of opinion
- Domain knowledge
- Infrastructure
- Involvement of multiple stakeholders
- Management policies
- Absence of a basic idea
- Required skill & resources
- Value addition
- Taking decisions from past failures
- Requirements based on future in mind
- Priority of requirements

The reasons for the requirement change could be endless. Some are valid and manageable requirement changes which could add business value. Such changes require expert handling given that the customer needs are a top priority. The changes, if they have to be implemented, should be a win-win for both the customer and the software developer. Proper change management practices will take care of such varied changes in requirements across the SDLC.



Multiple reasons for requirement change requests

CHALLENGES FROM TESTING PERSPECTIVE

As testers, we face enough challenges throughout the life-cycle of the project. Challenges come in many forms and sizes; challenges come during various times during the life cycle.

As there are challenges during the different phases of SDLC such as software requirements elicitation, analysis & design and coding, there is equal number of challenges or even more, during the testing phase.

The toughest challenge is the Requirement changes. Testers face this challenge from the day the requirements are gathered. Static testing is done on the requirements to find defects. The trouble starts when the requirement changes after a round of static testing is done on requirements. And this could just be the beginning.

Requirement changes could affect the following

- Testing scope
- Test plan
- Test strategy
- Test approach
- Test schedule
- Test analysis
- Test design
- Test scenario creation
- Test case preparation
- Test entry and exit criteria
- Test execution
- Testing effort & cost

The above list is indicative given that testing activities also include automation. In automation, we write functions and scripts and changes to requirements means the functions and scripts need to be revisited again thereby updating and tracking multiple times and multiple locations.

What ever changes that happen to the requirements, those changes have to be incorporated/updated/modified at all places involving the test documentation or test automation scripts and functions. These changes could result in

the entire plan for testing, the effort could be wasted, the milestones need to be modified and even the test execution could be halted or re-scheduled. These additional tasks are to be carried out in the interest of the quality of the product. Finally, a smallest requirement change could have a spiraling effect on the entire test life cycle.

REQUIREMENT CHANGES AND ITS IMPACT ON BUSINESS VALUE

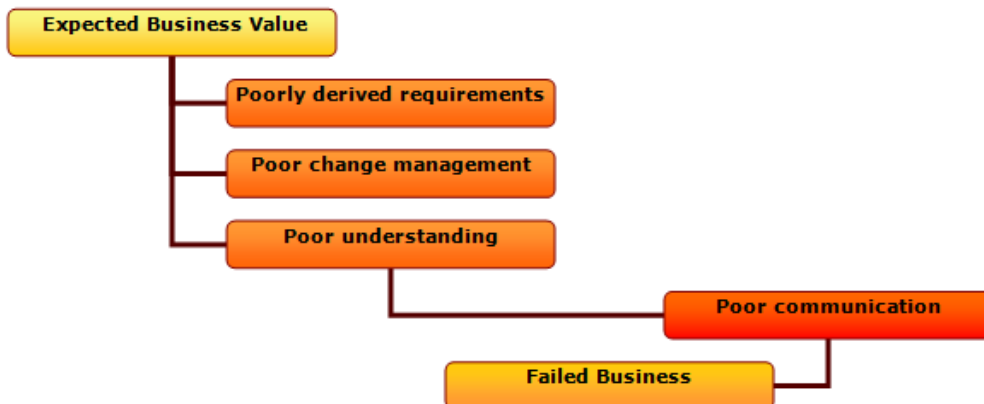
- Poorly derived requirements
- Poor change management
- Poor requirement understanding
- Poor communication

And other such derivatives of changing requirements have a huge impact on business value. Requirements are basis to the application being developed and the very future of the application lies on the correct requirements. The applications are built to get its benefits and add value to business, empowering users to do things quickly. But the user does not get the value as



Requirement change affecting the various phases of Software Test Life Cycle

expected. This is purely because of inadequate plans to handle changing requirements and its correct implementation. What ROI the customer expects from frequent requirement changes would indirectly hinder the value addition expected from such change.



Failed businesses could result in:

- Loss of credibility
- Loss of business to competitors
- Loss of skilled manpower
- Investment risk from partners and associates
- Dip in stock market
- Risk of bankruptcy
- Legal issues

REQUIREMENT CHANGES AND ITS IMPACT ON SOFTWARE QUALITY

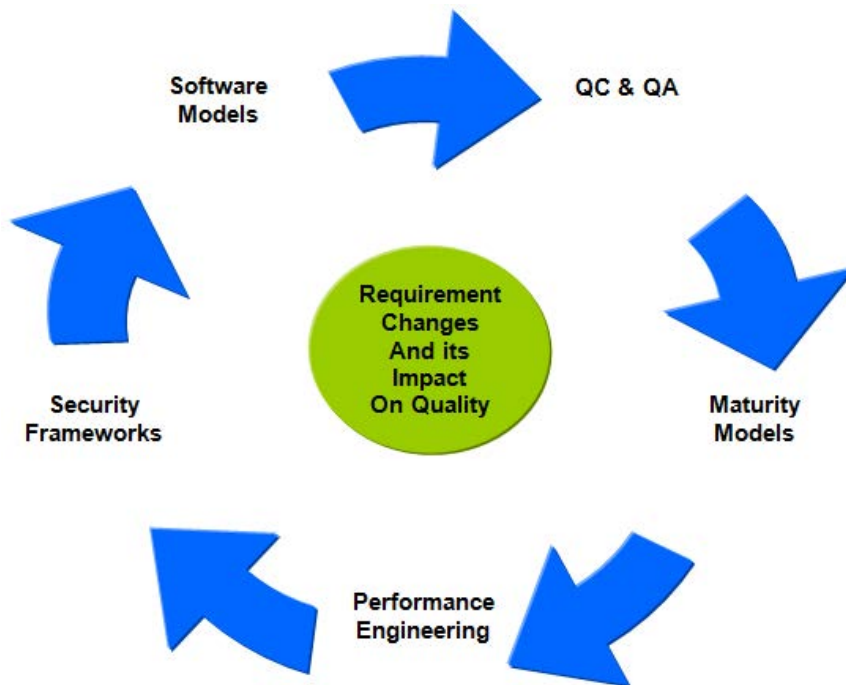
When the frequent requirement changes could lead to failed businesses, we can imagine the impact on the software quality. Perhaps, quality is impacted the most during the software life cycle. No one would risk their ROI on poor quality products.

There is a quality factor defined for software that is developed. It would directly or indirectly impact the quality if there is no clear focus.

- Understandability
- Completeness
- Conciseness
- Portability
- Consistency
- Maintainability
- Testability
- Usability
- Reliability
- Efficiency
- Security

These quality factors play a critical role in every software product that adheres to quality standards. Each one of these quality factors is dependent on the Requirements. And a frequent change to the requirements directly impacts the quality factors.

The processes, the models, the quality control and quality assurance initiatives, maturity models, performance and security frameworks etc. could also be impacted.



There has to be a process to handle proposed changes. If the requirements change often during the initial phase, then the impact would be minimal given that not much of effort would have gone into the STLC. Unless there are clear indications given to the stakeholders about the impending changes to requirements, nothing could be done later to avoid mishaps during the advanced stages of test life cycle.

The quality control and quality assurance initiatives would need a re-look in all aspects when a change takes place. A change could be big or a small one, periodic updating, tracking and managing the change and confirming it to the QC and QA processes are mandatory. There could be hidden challenges from quality perspective. The quality processes confirming to ISO standards or the maturity levels adher-

ing to the Capability Maturity Models (CMM) would have a hidden impact from the requirements angle. The impacted software quality has a direct link to the People, the Process and the Technology.

GOOD PRACTICES AND RECOMMENDATIONS

Good practices

Change requests are best avoided but it seldom happens as changes take place so often in the software life cycle. Whether we like it or not, we have to accept the fact that requirements often change. So far in the history of Software Industry, everything revolves around Requirements and it will continue that way.

Requirements traceability: Maintain requirements traceability, update and track periodically. Involving the customer would better for coordination.

Continuous Requirements management: Requirements have to be continuously managed in the way that it can be handled well, professionally.

Change management: This is the most common approach where the changes are tracked and managed. But, it may fail unless the changes are not gathered, analyzed, clarified and freed.

Release management: This is basically used for purpose of tracking the software releases but it also helps to know the planned and unplanned pattern of software being released or held back due to changes. It is useful when onshore-offshore kind of model is engaged.

Impact analysis: A thorough impact analysis has to be carried out to understand the pattern of requirements change. Such analysis could yield good results on the impact of the change on the application being developed. By doing an impact analysis, we are tried to know the impact of change on Design, Code, Software dependencies, Hardware dependencies, Network dependencies, Database dependencies, Project plan, Test plan, Configuration management plan, Release plan etc.

Recommendations

Though the good practices mentioned are good enough to face challenges on requirement changes, this paper tries to offer some recommendations. Although, every project is a unique challenge, some recommendations, if adopted, could help in handling the requirement changes in a way that could add value to the software life cycle.

Some recommendations that could help are:

- Change, once accepted, should be in phases only
- Change should be accepted and implemented based on priorities
- Change should be totally discouraged during coding or testing
- Accept critical changes with proper change management mechanism
- Accept change based on business impact
- Treat frequent change requests as new requirement and share effort and cost estimation
- Break or split requirements into smaller modules or functionalities
- Involve the customer when change occurs during testing phase
- Bring awareness into the team about the imminent change requests
- Statement of Work should be re-written based on the changes, if any
- The scope should be redefined keeping the cost implications in mind
- The SLA should be rewritten and agreed upon by both the parties before moving further
- All the stakeholders should be taken into confidence before implementing major changes

These recommendations are just indicative which may or may not work in all cases. However, an attempt should be made to streamline the change process.

CONCLUSION

It may take a while to say a polite NO to a customer who changes the requirements often, at least it should be attempted to find a way to educate the customer about the risks, consequences and negative impact on business and software quality.

REFERENCE

Software Requirements Engineering: What, Why, Who, When and How By Linda Westfall

http://en.wikipedia.org/wiki/IEEE_830

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

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

<http://www.processimpact.com/articles/reqtraps.html>

<http://ksi.cpsc.ucalgary.ca/KAW/KAW96/herlea/FINAL.html>

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

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

<http://software-qa-process.blogspot.com/2008/11/impact-analysis-checklist-for.html>

ABBREVIATION USED

SDLC Software Development Life Cycle

SLA Service Level Agreement

QA Quality Assurance

QC Quality Control

STLC Software Test Life Cycle

ROI Return on Investment

ISO International Organization for Standardization

CMM Capability Maturity Model



Rajesh Barde, a post graduate in computer applications is a passionate tester based out of Bangalore, India. He is an Independent Consultant in Software Testing, Test Process, STLC, Test Management, QA, QC, ISO and CMM. Additionally, he offers consultancy on Cyber Laws to Startups, Mid-sized companies and other corporates needing to have a policy on Cyber Laws. He is a blogger who writes at <http://bugburji.blogspot.com>. He has several whitepapers to his credit including a whitepaper published in IEEE xPlore and presented at an International Conference held in UAE.

Rajesh Barde tweets at @rrajeshbarde

Is It Time to Do Away with Low Level Test Scripts?

- Praveen Bhandari

I bet, either you have a smile on your face, or have your eye-brows risen, with this topic.

If not, let us start with - "What is a low level test script?"

Many a times QA folks divide their manual test script writing in two parts - High level scenarios and low level scripts.

High Level Scenario - Is a briefly described validation of a business scenario. For e.g. Test01: Validate that a user can Login with a valid username and valid password

Low Level Script - Detailed steps involved in execution of a test scenario, expected output and data used during execution, including boundary value analysis (max and min length of username, max and min length of password). For example - Test01: Open URL, Enter a valid username (max length), Enter a valid Password (max length), Click on Login button, Validate welcome message

Coming back to the question – Is it time to do away with low level test scripts?

Well, the answer is No and Yes.

No, because of the following limitations -

1. Test coverage - How can I be sure that I have covered all possible +ve and -ve functional validations
2. Test estimation - How would I gauge the efforts required in writing/executing one test script (usually no. of steps is a governing factor)

3. Domain knowledge - How can I be sure if the QA person has tested the complete functionality. Does he understand the domain or business function?
4. Knowledge Transfer - What if there is a new member in QA team, how would he know how to execute the test?
5. Developer's doubt - How would a developer know where did the test failed and caused a defect
6. Automation – How would an automation tester know what to automate?
7. Process Change – come on, I've been doing it this way for ages, why do I need to change?

And yes, because of the following amazing factors -

1. Improved efficiency – Less time spent on test script creation and more time on testing
2. High maintainability – changes in the application would need less maintenance activity on scripts
3. Adoptability – can be easily adapted to changing requirements
4. Creativity – Tester can think of multiple ways of testing a scenario and not just with the way a low level script is written
5. Faster defect discovery – improved defect discovery rate due to high speed execution

6. Improved business knowledge – Every tester would need to know the business functionality from end user's perspective
7. Efficient regression cycles – less time in execution and low time to market/live

Looking at the brief history of testing techniques there has been multiple interesting phases* –

Phase I. Before 1956: The Debugging-Oriented Period

– *Testing was not separated from debugging*

Phase II. 1957~78: The Demonstration-Oriented Period

– *Testing to make sure that the software satisfies its specification*

Phase III. 1979~82: The Destruction-Oriented Period

– *Testing to detect implementation faults*

Phase IV. 1983~87: The Evaluation-Oriented Period

– *Testing to detect faults in requirements and design as well as in implementation*

Phase V. Since 1988: The Prevention-Oriented Period

– *Testing to prevent faults in requirements, design, and implementation*

(*Source -

<http://www.cs.cmu.edu/~luluo/Courses/17939Report.pdf>)

The software development techniques have changed a lot in last decade or so where Agile and Rapid development have become more common than ever. With this pace, it is essential to look for efficient ways of testing. However, despite the numerous research results testing remains an - awkward, time -consuming, cost-ineffective chunk of work that is always not very satisfying in most industry practices.

It is always the case that testing ends up being a must-end activity because the project runs out of budget and is beyond deadline.



Praveen Bhandari, working with Sapiient Consulting, has over 12 years of QA experience in varied domains (healthcare, logistics, finance, e-commerce)/technologies/types and phases of testing. He started his

testing career with performance testing and over period of time got involved into full lifecycle of testing in delivery projects. He flairs for innovative methods of testing and values quality over quantity. He likes driving/travelling/photography and have recently started penning down his thoughts in forums.

International Society for Software Testing

ISST is a testing society for testers who are serious about their craft!

We're looking to change the testing world by promoting an approach to software testing that emphasizes value and the role that skilled testers play in its delivery.

And you are invited.

Join us!

<http://commonsensetesting.org/join>



BOOK WORM'S CORNER

September was amazing yet true. What was amazing yet true was that I got the same book referred while talking to multiple people. I am taking about "How to Solve it"; though I had an option to reach out to and ask for another book, we thought the book actually deserves two mentions due to the power of its content alone. Another amazing yet true factor was that it was not a testing book at all. Over to Karen Johnson and "a tester being" Jaysree Rathod below...

Love,
WoBo

BOOK RECOMMENDATIONS FROM THE TESTERS & WHAT THEY HAVE TO SAY ABOUT THEM

Karen N Johnson

How to Solve It by George Polya

Although this is not a testing book per se, I find this book has greatly helped my ability to solve problems. This is a book I go back to often, refer to other people and continue to find helpful.

Explore It! by Elisabeth Hendrickson

Although this is a new testing book, I can already tell that I'm going to be referring many people to this book. It is practical; it reads like Elisabeth talks and is easily "digestible" and understandable.

Jayshree Rathod

How to solve it - a new aspect of mathematical method by G. Polya

As the name suggests, it helps us getting close to the solution of the present problems. It shows the path to reach to "that" unknown by identifying patterns, use of available data, solving similar problems or solving the part of the problems.

Exploring requirements - Quality before design by Donald C. Gause & Gerald M. Weinberg

From start to the end it focuses on exploring and understanding the requirements process, how we can reduce ambiguity, and limit the expectations. A very good book for anyone who deals with the products.

The Simplest Web Automation Tool



Simple

Powerful

Productive

Simple powerful scripting

Smart Object identification - No complex Xpaths

Powerful record and playback -Any Browser Any OS

Fast parallel batch playback

Ajax? No time out issues

Become our fan -

https://twitter.com/_sahi

<http://www.facebook.com/sahi.software>



Request a free demo by sending us an email at support@sahi.co.in

<http://www.sahi.co.in>



A Fake Tester's Diary

<http://www.testingcircus.com/category/a-fake-testers-diary/>

The Protector

Sometimes, undetected bugs are good for the product; an incident that did not occur in software was during the making of the James Bond movie "Tomorrow never Dies"; The movie was originally titled "Tomorrow never Lies"; a producer saw a printing mistake on a poster which read "Tomorrow Never Dies" and decided that it was more apt for the movie. Likewise, sometimes when you see a bug, try to analyze if it's a good bug for the product and if it is, advise your teams to launch along with the bug instead of getting into a battle asking someone to fix the bug.

September is the beginning of the festival season in India; most people will have at least 1 occasion in the next few months when they can pray to their favorite gods --- for peace, prosperity and for protecting themselves. In the corporate world, with multiple people testing and with the speed at which builds land up, it's highly likely that you might have missed a bug and someone else would have found it. It's even more likely that they use this bug as a bellwether to pass judgment on the testing team. As testers, there are multiple times when the focus would be on you to respond as to why you did not find out about a bug (which someone else did); as an alternative to praying to god, here's presenting you a checklist of 11 items from which you can choose the most appropriate one to protect yourself. You can use any of these options when your boss or client or manager or senior manager or super-senior manager asks you why you did not find this bug and someone else did.

If the question is "Why did you not find out about this bug wherein the user is unable to login with the testuser2 login ID", the answer can be.....

1) Not in Scope --- Testing this scenario is not in the scope since this scenario is not covered by the requirements document

This is the most common answer; you should know that everybody would have forgotten that the "Requirement Document (Business Requirements Doc or any such named doc)" exist only to give guidelines to the engineering teams. When questioned, feel free to refer to the requirements document and check if it has the specific scenario mentioned and if it is not present, use it as a very effective defense mechanism to state why you did not find out the bug earlier.

2) That feature was tested in a prior build and hence not tested in this build.

In case the feature is present in the requirements, then you are out caught; to save yourself, you can say that it was tested in build 1.00.343-0878A and the release build was 1.00.343-0878B. Feel free to blame it on the lack of a process that does not catch last minute irresponsible check-ins and feel free to blame it on communication. This excuse is very effective for internal projects, but I don't think you can use it for external client projects.

3) Oh, I thought it was already tested by the automation teams.

In case the above 2 reasons are irrelevant, then you can blame it on the automation teams; say that you tried to do "smart testing" by not doing what the automation teams were doing and that you thought that this was supposed to be done by the automation teams. Blame it on the "agile" process if people ask you for meeting notes, and documentation to confirm that the automation teams were doing it. Put the pressure to the automation teams to state why they did not actually catch this bug using their scripts? Transfer the irresponsibility & blame to them so that you are safe.

4) This was supposed to be caught by a different testing team and not by our team.

When you have multiple upstream and downstream systems, you can easily get away stating "integration issues"; you can say that you are unaware of what happens in downstream and upstream systems; include the factor of "24 hour testing" so that people will understand your long working hours; feel free to pull in the project management teams since they did not keep you apprised of what's happening downstream and upstream. This will delight your "CMMi" and "6-Sigma" people since they will use this lack of process to try and sell their "so-called effective process models (or defective, as I call it)" to senior management.

5) Our test case asks us to test with username of "testuser1"; this bug occurs with a username of "testuser2".

If you are billing your client by the hour, and he has actually seen your earlier set of test cases, feel free to use the Sign-Off factor. Claim and reclaim that all these tests were actually "signed off" by your client and hence, you were not testing these scenarios; your management would love this excuse since that would set them free. Ask your onsite coordinator to include a "Change request", "New Requirement" or whatever name works and in all future cycles, charge your client for this.

6) Oh, this is not mentioned to be tested according to the latest minutes of meeting of this project with the customer, the minutes which has been stored in that server which is inaccessible to most of us.

If you think that this change is related to a recent modification in the project, feel free to put the blame to the process and documentation associated with this change; this can save you on your 1st goof-up if you are new to the team.

7) The test lead/Test Manager/Test Super-Manager has not approved us to test for this specific item.

And if you think that it is revenge time for you, point out your immediate supervisor as the sole root cause of this damage; state that he has not provided approval to test this specific aspect of the application; you can do this if you are planning to change teams, or if your supervisor is in the process of changing teams.

8) Oh, I logged this long back; it was prioritized as a low priority one.

And yes; if you have some evidence for you having witnessed the bug earlier and logged it in the bug-tool or any obscure email from earlier, flag it for all to see and state how you had actually identified the bug much earlier; this is an effective tool to save yourself and you can even use this to showcase how you know more about the project than anybody else.

9) Hmmm, we are working as per CMMi guidelines; this is not a bug according to the CMMi guidelines.

Living in a world with multiple process models like CMM5i or 6-Sigma, you can try to state a case that this scenario need not be tested on that specific process model that you follow while you are testing. Call in the process expert to fix the process bugs and he will be happy to get involved!

10) That is a bug we would have found out if we did "low-level" sanity testing; but we did "high-level" smoke testing and did not find it.

Feel free to make use of words like "low-level", "high-level", "medium-level"; feel free to use more combinations like "low high-level" and "high medium-level"; suffix them with words like "sanity testing", "smoke testing", "BAT testing", "BVT testing". Create at least 1 full page with such combinations and end it with something like "Next time onwards, we will do 'Planned Exploratory Deep Smoke Testing' to find it". Nobody would question you when you post such definitions!

11) Ah, this does not occur on my machine.

And this one is an all-time classic; feel free to use this whenever you run out of excuses; it's not that only the developers are allowed to use this excuse; testers can, too!

What is very important in your response would be the "Ooh" or "Ah" or "Umm" or "anything like that" before your response; that would give the false sense of notion that you are actually an expert in this product and the person will not ask additional questions.

Of course, the true answer to the original question would be

- I missed it.

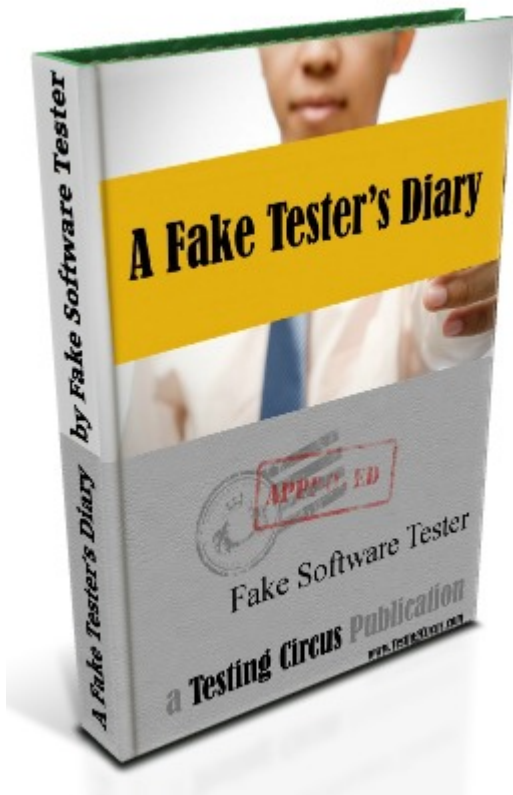
- I did not understand the context of the project to think that this scenario has great implications on the client.

However, the truth will land you in grave danger and give your bosses sufficient ammunition when you ask about that raise or promotion; Corporate teams love each of the 11 excuses, especially when they can pass them on to others as well and hence, feel free to use

them ~~at your peril~~... um, at your convenience! These will ensure that you don't run out of protective options for at least a year. If you run out of them, well feel free to write to me :) ... Happy Protections!

Read more fake tester's diary at -

<http://www.testingcircus.com/fake-tester/a-fake-testers-diary>



New readers are encouraged to read our old editions of Testing Circus.

The eBook is available in PDF, ePub and Mobi format. It can be purchased from

https://leanpub.com/FST_TestingCircus

Our subscribers will get it free. The 100% discount code has been emailed to all our subscribers.

Subscribe now at

<http://www.testingcircus.com/subscribe> to get it Free.

WeTest Weekend Workshops

Saturday, November 30, 2013

New Zealand has a wide culture of development led weekend events, but until recently there was little out there for testers who wanted to develop their craft. A major influence within Wellington has been the emergence of WeTest peer conferences, which have provided a great forum for people to network and share their experiences. However peer conferences can be daunting for those new to them, and indeed numbers tend to be limited. It soon became obvious that there was an appetite for an event which could be inclusive but affordable. The WeTest Weekend workshops - sponsored by consultancy Assurity - are an affordable mixture of an afternoon of discussions and workshops around testing led by testers representing the many IT companies from around Wellington. Their aim is to explore and challenge ideas and techniques whilst providing a forum for networking within the Wellington test community.

Registration closes Thursday October 31st and applicants will be advised of their conference programme in early November.

The list of workshops on offer are:

- Changing an organisation one elephant at a time [BRIAN OSMAN]
- Exploring and Testing Gravity [MIKE TALKS & DAVID ROBINSON]
- How to test a web service [NOEL DYKES]
- Specification by Example [NIGEL CHARMAN & AMY LANGRIDGE]
- Test Manager - Who am I? [ANNA MARSHALL]
- Testing mobile and your sanity [JONATHAN ELKIN & SAM MEIKLE]
- Thoughts on test automation [CHRIS ROLLS]
- Is this really how it's supposed to be done? [CRAIG MCKIRDY]
- Rapid Testing Challenge [SARAH BURGESS & DAMIAN GLENNY]
- Trial by Reason - Challenge the Logic! [JOSHUA RAINE]
- The Session Obsession [ADAM HOWARD & SAMANTHA NIX]
- Agile Testing - are we doing it right? [ED FELIX]



<http://www.meetup.com/WeTest-Workshops/events/138808582/>



We recommend to follow

Bernice Niel Ruhland

Software #Testing Manager. Regular article contributor of Testing Circus magazine. I have #cooking blog.
<http://realisticcookingideas.com> Trekkie! Love Disc Golf!!

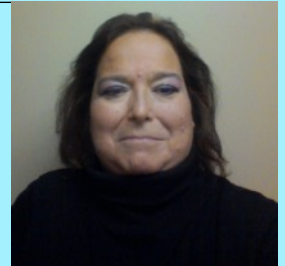
<https://twitter.com/bruchland2000>



JeanAnn Harrison

Software Tester, Mobile Testing, Quality Assurance Engr, Mobile Marketing Strategist, Sports Fanatic, Adult Competitive Figure Skater.

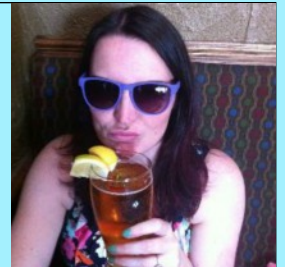
https://twitter.com/JA_Harrison



Julie Hurst

Lead Tester for Digital Marketing at FGL Sports | Gen Y | Columnist | Runner | Skier | Australian | Scotch Drinker | Chocolate Addict.

<https://twitter.com/joolery>



Susanne

Software Tester @swissq; Mobile Testing, Requirements Engineering, Usability - "Enjoy life. There's plenty of time to be dead."

<https://twitter.com/suezzun>



<http://Twitter.com/TestingCircus>

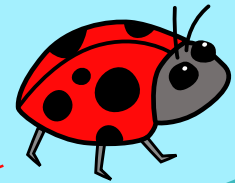


Twitter

Testing Circus

Software Testing Magazine. Founded and Edited by @ajaysingha.
First Publication September 2010. Monthly publication.
Subscribe Free at <http://testingcircus.com>

Follow us at <http://twitter.com/TestingCircus>



Follow us at Twitter

<http://Twitter.com/TestingCircus>



Crack The Code!



- **Blindu Eusebiu**



As a bug hunter, you need to have the proper environment to receive the money/reward. For example Paypal bug bounty pays via a Paypal verified account. If you don't have a Paypal verified account you would avoid the program and miss some rewards. The same with taxes. I suggest you pay the taxes and have everything ok. Because if you don't pay the taxes it will worry you, you will be subconsciously avoiding to get more money because it will be obvious and you will be afraid of getting caught.

The puzzle is: list as many bug bounty programs with their methods of payments.



Eusebiu Blindu has been working as a tester since 2005. He is experienced in working in a couple of companies and environments like IBM or European Commission. In the most recent years, he has tried to find more meaning in the activity of testing and how it could be more motivating, more rewarding to be a tester. He has experimented with different schools of testing, like context driven and different approaches like exploratory testing. But he didn't settle for any group or methodology as he thinks all of them have their own flaws. After discovering uTest, he is focused more on security testing and security

bug bounties.

He tweets as @testalways.

You can find more interactive testing puzzles on his website <http://www.testalways.com>

Send your answers to editor@testingcircus.com

JAY PHILIPS

Organization: Project Realms, Inc. & TeamQualityPro

Current Role/Designation – CEO & President

Location – Minnesota

Interview with Testers

Jay Philips is a highly experienced leader with a focus in business intelligence and software development team building.

She has successfully completed multiple large and small projects in a variety of industry domains including strict regulated financial & health care environments.

Over the years, she has concentrated on learning more about the entire software life cycle, quality process, and controls for a project. She is familiar with a variety of testing tools which support a full software and system life cycle including requirements tracking, configuration management, defect tracking, test team management and automated regression test tools. In doing so, she has used metrics as a means to measure a project's success.



Interviewed by Ajoy Kumar Singha

1. The usual question. Tell us about your journey to becoming a software tester. How did it start and how this has been so far? Was it planned or by accident?

I originally started as a Network Administrator - I never knew software testing was even a field. I was at a client site setting up their authentication system when I found a very critical bug within an application that they had planned to go live with that weekend. While the client had a test team, the issue had gone undetected. I was asked to stay on and help with testing the application until all the critical issues were resolved. Not only did I stay on, testing became my passion because it felt great knowing that I was finding bugs before the end user did.

2. Do you regret being associated with software testing today? Given a chance would you move from testing to any other field in IT or any other non-IT domain?

I do not regret it at all – I love it! I actually owe it all to that one critical bug I found more than a decade ago or I wouldn't be where I am today.

3. You founded Project Realms in 2009. Was it because there was a need for such type consulting firms? How did it start and how are response so far?

I founded Project Realms so that I could put the client first. Many of the firms I know of are more about their own bottom-line and are not necessarily about the success of their clients. I also noticed that they didn't focus in on the entire project lifecycle, which is what I believed was really needed. We have had a great response on how we conduct and manage projects from start to finish. We have experienced significant growth each year based on our approach and commitment.

4. You recently released the TQP platform. How is it different from other dashboards that we see in the market?

TeamQualityPro (TQP) is currently the only dashboard system that allows a user to see the entire application development organization. Other dashboards focus on singular items such as: defects, test scripts, etc. Therefore, the only way to get a full view, if you're not using the same tool, is to export to excel and create metrics manually. TQP is code agnostic, built on lean architecture, automated and



mobile ready so you can get updates anytime, anywhere. The product has been very well received.

5. You recently attended CAST 2013. What was your take away?

As always, CAST was a wonderful conference. I am on the Leadership SIG, so my biggest take away is to start getting more traction on the SIG.

6. How are you associating with other testers around the world apart from attending testing conferences?

I am very active on Twitter and Facebook, so I am able to keep connected with other great testers all the time.

7. According to you, what is lacking in today's commercialized training industry, especially in testing?

Most colleges are still not offering stand-alone courses in software testing. I think more people would come into the field if they were told about it. The AST Education SIG is currently working on a SummerQamp, which will help get high school students more involved but I think it needs to go even further into private and public colleges.

8. What qualities will you look for in a candidate when you want to recruit someone for software testing job?

The ability to think outside of the box. There are so many testers that will just follow a detailed test plan or do point & click scripting. I wish more testers were more context-driven.

9. What will you suggest to people who want to join IT industry as software testers?

I wish there was a course I could recommend but instead I'd say spend some time reviewing some of the testing blogs & videos, join one of the

conferences to get a better feel of what software testing is, and job-shadow with a tester you know.

10. Name few people you would like to thank, people who helped you directly or indirectly in your career as a software testing professional.

I thank my husband for putting up with my very long hours and crazy travel schedule, the entire Project Realms & TQP team for joining me on the path we're creating, and all our clients for being wonderful partners.

11. Jay Philips minus testing is – This is a tough one since it's my core background.

I guess I'd be a Project Manager. Project management is a great field but it is just not as much fun if you do not have testing experience.

12. One last question – Do you read Testing Circus Magazine? If yes, what is your feedback to improve this magazine?

Yes, I do. I think a section on what I learned from a previous magazine article would be great.

Blog/Site – <http://www.projectrealms.com>

<http://www.teamqualitypro.com>

Twitter ID – @jayphilips

Promoting Software Testing



Free eBook

Rob van Steenbergen's ebook Promoting Software Testing in Your Organization is available for download from Testing Circus site.

Some highlights from the book

- Create a Communication Strategy
- Promote Testing when You are the First Tester in the Organization
- Don't have a Bugs Database yet?
- Tell People about the Benefits of Starting Early with Testing
- Do Regular Product Risk Analysis
- Have an Elevator Pitch Ready at Hand
- Create a Recognizable Logo for Your Test Team
- Go to Test Events, Take Your Colleagues Along
- Publish in the Organization's Magazine about Testing
- Stay Positive about Testing
- If Someone asks You to Give a Presentation on Testing: Always say YES!

The suggested hashtag for this book is [#PromoteTesting](#)





Issued in Public Interest by
Test Mile Software and Services

Together. We Cover.

Divided. We UnCover.

Development

Testing

Security Testing Tips

Part 9

- Blindu Eusebiu

Web Security Bug Bounty Cheat Sheet

Domains in Scope	Find domains in scope	<ul style="list-style-type: none">• Read the bug bounty specifications• Explore the platform to find domains that belong to it• Check if acquisitions are in scope and find the domains for every acquisition• Check public domain listings for the same company eg. 'domaintools.com'• See what other valid domains were accepted by reading bug hunters blogs• Google
Sub-domains	Find sub-domains in scope	<ul style="list-style-type: none">• Run a basic sub-domain scanner that uses DNS records• Find sub-domains by exploring sub-domains already known (in www.domain.com there could be a link to ftp.domain.com)• Run a brute force scanner like knock• See other people' blogs• Google (eg search 'site:domain.com')
Directories	Detect sub - applications	<ul style="list-style-type: none">• Test per directories (look first in 'www.domain.com/contact', then 'www.domain.com/files' etc)• Use a scanner like 'Dirbuster'• Learn the functionality so that you can develop better attacks
Mobile platform	Find domains/sub-domains	<ul style="list-style-type: none">• The mobile platform is worth to check even if mobile bugs are not rewardable because you can find new sub-domains (mobile.domain.com)
Technical approach	<ul style="list-style-type: none">• Typical bugs• Exploratory approach• Use scanners• Read other bug reports	<ul style="list-style-type: none">• Search for bugs that you know are rewarded 'xss', 'CSRF' etc• Spend time in exploring to find known and atypical bugs• Use scanners that work for you best, the scanners that others recommend• Learn from others (blogs, articles - bug already rewarded, tools, tips etc)
Statistical/decision approach	Decide where to look	<ul style="list-style-type: none">• Chances are higher to find bugs where you already spent time and was rewarded for bugs• Look in areas where others reported bugs• Focus your approach based on your target (income, learning etc) - sending more small bugs is more recommended for income
	State of mind	<ul style="list-style-type: none">• Look for motivation to find bugs - why do you do it (fun, money)• Be patient!
	Stay updated	Read about new bug bounty programs, what others reported, how credible is the program etc
	New/Old bug bounties	Chances to find rewardable bugs are higher in new bug bounty programs

*** Blindu Eusebiu is a tester for more than 5 years. He considers himself a context-driven follower and he is a fan of exploratory testing. He tweets as @testalways.



Developer

Just deployed new build. Yahooooo!

6 hours ago Like · Comment

Project Manager, Lead Developer likes this.



Test Lead Go on testing guys. Attack, destroy the build. :P
6 hours ago Like



Tester I reported 102 bugs. Happy weekend! LOL.
6 hours ago Like



Developer @Tester Bug Id #87 - It is not a bug. It is working in my machine. Please check.
6 hours ago Like



Tester I double checked. It is a bug. Which browser are you using?
6 hours ago Like



Developer IE 6 in my machine.
6 hours ago Like



Test Lead Holy shit. You are still using IE 6? That too IE. Get a life.
6 hours ago Like



Tester Ha ha ha @Developer, your browser is as old your coding skill.
6 hours ago Like



Project Manager Guys, finish this asap. There is a dead line.
6 hours ago Like



Lead Developer Deadline my There are 201 bugs to fix.
6 hours ago Like



HR Use of such language is not allowed in this organization.
6 hours ago Like



Barack Obama Send all the bugs to USA. We'll fix 'em all. Yes, we can.
6 hours ago Like



Osama Bin Laden I can help you how to hid the bugs. you Obama. Go lick Syria's




Tester You guys carry on. I am going home, time to enjoy weekend and read "Testing Circus".
6 hours ago Like

+ Add a comment...



Lots happening in our Facebook Page too.

<http://www.facebook.com/TestingCircus>



LEARN SAHI STEP BY STEP

Sahi Tool Tutorials - brought to you by Sahi Team



AUTOMATING FILE UPLOAD USING SAHI



Tutorial - 7

- Narayan Raman

Sahi Pro provides support for automation of flex applications via the Sahi Flex Library (SFL).

To enable automation, the flex application needs to be compiled with Sahi's swc file.

Different swc files are available for different versions of Flex.

Compile your application with the relevant swc file.

If your application is compiled with Flex 4.0, use sfl4.swc

Sahi's swc files are located in `sahi_pro/sfl/` folder

Compile with sfl.swc using command line

You can compile your flex application with sfl using the following command (Change sfl version as needed)

```
mxmmlc yourapp.mxml -include-libraries+=sfl4.swc --output=yourapp.swf
```

After compilation refresh the browser cache, to make sure that the modified app is available.

Compile using Ant

When compiling using Ant, we need to specify the `compiler.include-libraries` option

```
<mxmmlc file="sample.mxml">  
... normal options  
<compiler.include-libraries dir="D:/sahi/sfl" append="true">  
  <include name="sfl4.swc" />  
</compiler.include-libraries>  
</mxmmlc>
```

After compilation refresh the browser cache, to make sure that the modified app is available.

More details on flex compilation itself via ant is available [here](#)

Compile using Adobe Flash Builder (Add SWC files to Flex Builder projects)

1. In Flex Builder, select your Flex project in the Navigator.
2. Select Project > Properties. The Properties dialog box appears.
3. Select Flex Compiler in the tree to the left. The Flex Compiler properties panel appears.
4. In the "Additional compiler arguments" field, enter the following command:

```
-include-libraries "sfl4.swc"
```

In Flex Builder, the entries in the `include-libraries` compiler option are relative to the Flex Builder installation directory.

The default location of this directory on Windows 32 bit is

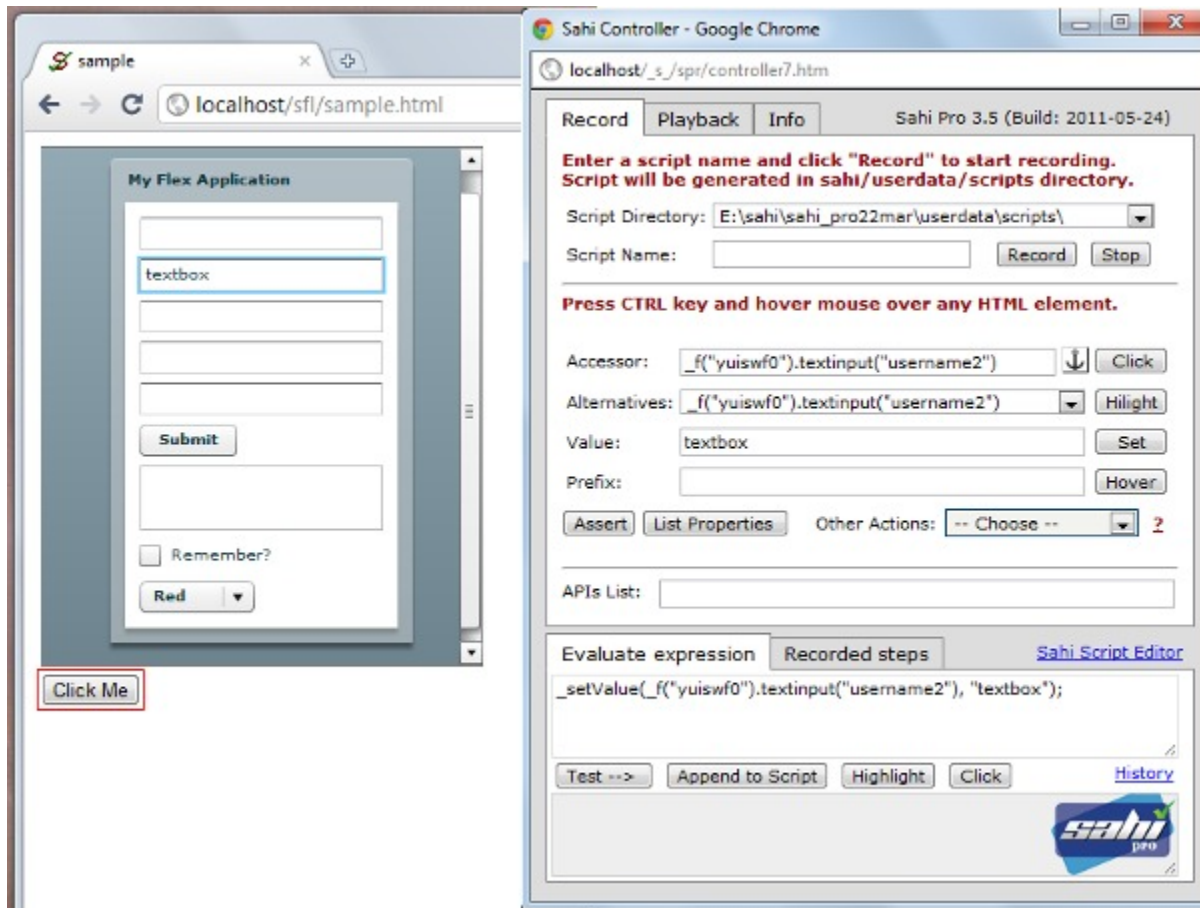
C:\Program Files\Adobe\Flex Builder

and on Windows 64 bit is

C:\Program Files(x86)\Adobe\Flex Builder

After compilation refresh the browser cache, to make sure that the modified app is available.

Recording a flex application



From the dashboard, open any browser and navigate to your flex application. Press ALT and double click on the document window of the page which you want to record. Sahi's Controller window will popup. You can now start recording your Flex application. Ctrl + hover (to get a flex element's accessor) will work only if the Flex application is in focus. To do this, you

will have to first click on the Flex application.

Flex APIs are slightly different from the normal JavaScript APIs.

Example

```
_f("flex_app_id").textInput("username2")
```

The code above identifies a `mx.controls::TextInput` field inside a flex application identified by `_f("flex_app_id")`

Identification of Flex Components

Sahi identifies flex components based on the type (class name) of element and one of its visible/significant attributes.

The attributes may be one of `label`, `text`, `name`, `automationName`, `toolTip`, `id`, `autoGeneratedName`, `index` or some other attribute which is relevant to that element.

mx components are identified as lower case of their class names.

Eg. `mx.controls::TextInput` corresponds to `API _f("id").textinput("id2")`

Spark components are identified with an `s_` prefix.

Eg. `spark.components::TextArea` corresponds to `API _f("id").s_textarea("id2")`

Recognizing Custom Components

Sahi, by default, adds support for most mx and spark components.

If your component is not recognized, do the following:

1. CTRL Hover on the element which is a parent of the element you want to recognize. You can use up/down arrows near the Accessor field on the Controller.
2. Once a parent element is identified, click on the "List Properties" button on the Controller. It will list down all components inside the parent and also all attributes.
3. In the fully qualified classnames visible, make an intelligent guess to narrow down your element of interest.

Suppose your class name of interest is `my.package::MyClass`

To enable Sahi to recognize custom components in your application, edit `sahi_pro/userdata/config/user_extensions.js` and add,

```
SflWrapper.prototype.addCustomMetaData = function(){
    this.addMetaData({qn: "my.package::MyClass",
        attributes: ["label", "text", "name", "automationName", "toolTip", "id", "autoGeneratedName", "index"],
        action: "click", value: "label", apiName: "myotherclass"});
}
```

In the above,

1. change `my.package::MyClass` to your relevant class
2. For clickable elements, keep `action:"click"`. For text input elements, use `action:"setValue"`
3. Set `value` to the attribute which conveys the value of that field. For example, a `dateField` will have `selectedDate` as its value attribute. This will be visible in the "Value" field on the Controller, and is used in assertions.
4. The name of the API is by default the lowercase of the classname. If `apiName` attribute is specified, that is used instead. Eg. in the above case, the api becomes `_flex("id").myotherclass("id2")`. If `apiName` was omitted, the api would have been `_flex("id").myclass("id2")`.

Performing actions on Flex elements

Sahi's Action APIs also work with Flex elements.

Examples:

```
_click(_f("mxComponents").button("ColumnChart"));
_setValue(_f("mxComponents").textinput("txt1"), "English");
_setValue(_f("sparkComponents").s_richeditabletext("richEdTxt"), "Hi there!");
_setSelected(_f("sparkComponents").s_combobox(0), "Cucumber");
```


Relational operators in SFL

SFL supports near, inside, leftOf, rightOf and under APIs.

The usage of these APIs is a little different from the Sahi relation APIs.

1. The relation APIs are called as member functions.

```
_f("fid").textinput("u").near(_f("fid").label("User"))  
_f("mxComponent").datagriditemrenderer("/@fictitious.com/").rightOf(_f("mxComponents").datagriditemrenderer("Mary Jones"))
```

2. They can be chained to use multiple relations.

```
_f("fid").textinput("u").near(_f("fid").label("User")).inside(_f("fid").vgroup("SettingsPanelView"))
```

Known Issues

1. For the CTRL-Hover to work, one needs to click on the Flex app and bring it in focus.
2. File upload/download are not currently supported.
3. NumericStepper recording does not work. However it can be identified from the Controller and scripted.

Tips on usage

The flex recorder in Sahi is not as sophisticated as the web recorder.

To introspect and identify elements, one can use the Evaluate Expression box.

Ctrl Hover on any Flex element. It will populate something like

```
_f("sampleId1").textinput("username")
```

1. One can see all attributes of the textinput field by one of the following

1. CTRL Hover on element and Click on List Properties
2. Copy the accessor into Evaluate Expression box and click "Test->"
3. Explicitly put component.listProperties() into Evaluate Expression box and click "Test->"
`_f("sampleId1").textinput("username").listProperties()`

2. To fetch the value of an element, use `_getValue()` API.

Depending on the type of element, Sahi will correctly return its "value" attribute.

```
_assertEqual("abcd", _getValue(_f("mxComponents").textarea("textarea")));
```

3. To fetch the text of an element, use `_getText()` API.

On flex it fetches the textual content of the component (behaves like `innerText` in HTML).

```
_assertEqual("abcd", _getText(_f("mxComponents").textarea("textarea")));
```

4. To fetch any property of an element, use `.get`

Example

```
_f("sampleId1").textinput("username").get("text")
```

or

```
_f("sampleId1").textinput("username").get("label")
```

5. To execute any member function of an element, use `.executeFn(arg1, arg2 ...)`

Example

```
_f("sampleId1").popupbutton("menu").executeFn("open");
```

or

```
var $fontSize = _fetch(_f("sampleId1").textinput("username").executeFn("getStyle", "fontSize"));
```

6. To see which Flex components make up a particular component, use introspect()

Example

```
_f("flexId").textInput("tId").introspect()
```

This will show a list of all elements inside the textinput field.

The output may look something like:

```
textInput0>Panel4>username>HaloBorder17:mx.skins.halo::HaloBorder
textInput0>Panel4>username>UITextField8:mx.core::UITextField #abcd
```

7. To see all elements in a flex component, (especially when an element is not identified) use

```
_f("sampleId1").introspect()
```

8. To check if an element exists, use `_exists` or `.exists()`

```
_assert(_exists(_f("mymovie").datagrid(0)));
```

```
//or
```

```
_assert(_f("mymovie").datagrid(0).exists());
```

9. To check if an element is visible, use `_isVisible` or `.isVisible()`

```
_assert(_isVisible(_f("mymovie").datagrid(0)));
```

```
//or
```

```
_assert(_f("mymovie").datagrid(0).isVisible());
```

10. To get data from a grid, use `.getGridData()`

```
_assertEqual([[ "a", "a1"],["b", "b1"]], _f("mymovie").datagrid(0).getGridData());
```

11. If `getGridData` does not work or if we are dealing with a graph, it is easier to get the `dataProvider` of the element and verify that the data is correct.

To get the `dataProvider`, do

```
_f("mymovie").datagrid(0).get("dataProvider")
```

This will return a JSON string. To get an object out of it, use

```
var obj = eval("(" + _f("mymovie").datagrid(0).get("dataProvider") + ")");
```

`obj` will now be an array of objects, which looks something like this:

```
[{"balance":"1000","category":"Personel Care","date":"15/12/2010","description":"ATM Fee",
"price":"1000","quantity":"1000","payee":"ABC","amount":"1000","datePaid":"15/12/2010",
"status":"success","action":"buy"}]
```

12. SFL supports indexes and regular expressions just like normal Sahi APIs.

For example, to look for `maurice@fictitious.com` we could do

```
// returns maurice@fictitious.com
```

```
_getText(_f("mxComponents").datagriditemrenderer("/@fictitious.com/[2]"));
```

```
//using relation and regular expression
```

```
//returns maurice@fictitious.com
```

```
_getText(_f("mxComponents").datagriditemrenderer("/@fictitious.com/"))
```

```
.rightOf(_f("mxComponents").datagriditemrenderer("Maurice Smith"))
```

Flex app and SahiPro version

Your Flex app needs to be compiled with the appropriate swc file based on the Flex SDK used.

It does not have a dependence on the Sahi Pro version itself.

The swc files available for compilation can be found inside the `sahi_pro/sfl` folder.

CONNECT WITH SAHI TEAM -

https://twitter.com/_sahi

<http://www.facebook.com/sahi.software>



Narayan Raman is the founder, CEO of Tyto Software Pvt. Ltd., a Bengaluru based software products company specializing in software automation products. Narayan is the author of Sahi, an award winning open source web test automation tool. He has over 13 years of experience in the industry and holds a B. Tech degree in Chemical Engineering from IIT Bombay. Tyto Software, started in 2008, helps organizations simplify and achieve success in their test automation process. Tyto is now a small, successfully growing company.

Testing Circus

Testers' Knowledge Network

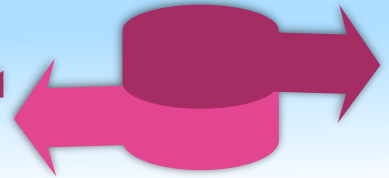
Register on our site and start creating profiles, post messages, make connections with testers, create and interact in groups, download monthly magazine, read testing articles and much more.

Connect Now



Looking forward to something cool? Register here -
<http://www.testingcircus.com/register>

QTP Code Corner



Working with Multiple Browsers in QTP

- Naina Dhiman

Some times in a project we have to handle multiple browsers which are opened as a result of some particular steps.

NOTE: We will be using Descriptive Programming here.

This can be handled by Ordinal identifiers such as **Index and Creation time.**

Index: Index value is based on the order in which the object appears within the source code. The first occurrence has a value of 0. The Index property values are objecting specific.

Creationtime: This identifier is only available in Web Environment for Browsers only. The value of this identifier indicates the order in which the browser was opened relative to other open browsers. The first occurrence value is 0 and then for 2nd browser is 1 and so on.

For example:

Script 1:

```
SystemUtil.Run "iexplore.exe"  
Browser("micclass:=Browser").Navigate  
"www.google.com"
```

Here in highlighted statement, only one single browser will be open. Now let modify this script:

Script 2:

```
SystemUtil.Run "iexplore.exe"  
SystemUtil.Run "iexplore.exe"  
Wait(2)  
Browser("micclass:=Browser").Navigate  
"www.google.com"
```

The highlighted line will give an error as 2 browsers are open here. To solve this problem we use INDEX or CREATIONTIME ordinal identifiers.

To refer to 1st Browser:

```
Browser("micclass:=Browser", "index:=0").Navigate  
www.google.com
```

Or

```
Browser("micclass:=Browser",  
"creationtime:=0").Navigate "www.gmail.com"  
Both the statement will produce same result.
```

To refer to 2nd Browser:

```
Browser("micclass:=Browser", "index:=1").Navigate  
"www.google.com"
```

Or

```
Browser("micclass:=Browser",  
"creationtime:=1").Navigate "www.google.com"  
Both the statement will produce same result.
```

Browser Identification Issues:

Use of Index or Creationtime property may cause some problems in some situations like :

- 1) If before running the script there are one or more browsers already open.
- 2) If the script is unable to close the browsers it spawned in subsequent iterations
- 3) While running script from QC. In this situation the QC browser can be treated as one of the browser and is close by script.

Some ways to avoid above situations:

- 1) Close all Internet explorer processes before launching any browser or running a script.

Following statement can be used:

```
SystemUtil.CloseProcessByName "iexplore.exe"
```

It will close all the existing IE instances.

Note: This statement is not helpful if we are running our scripts in QC.

2) In case, if we are using Browser property in Object Repository with defined Creationtime, we can use SetTOProperty to set the creation time at run-time also. (SMART identification should be disabled to make this work properly.)

```
Browser("Browser").SetTOProperty "creationtime", 2
```

In addition to the above methods, there are also some Browser Identification Properties listed as below:

Browser Identification using OpenTitle Property:

The browser's Opentitle property defines the browser's initial title when it is first launched:

```
'define the description String for the browser  
dpBrowser= "OpenTitle:=Google "  
SystemUtil.Run "iexplore.exe", "www.google.com"  
Browser(dpBrowser).Navigate "www.gmail.com"
```

Note: Here, we should know the page title corresponding to the URL. Also running the above code twice will result in error because then there will be 2 browsers with same OpenTitle property.

Browser Identification using a Unique OpenURL Property:

The following method specifies a junk with unique value.

```
' Generate a random URL every time  
browserID= RandomNumber.Value(10000,99999)  
' Get the open url String  
dpBrowser="OpenURL:=about: " & browserID  
  
' Launch a new browser with the opening URL based  
on random number  
SystemUtil.Run "iexplore.exe", "about: " & browserID  
Browser(dpBrowser).Navigate "www.google.com"
```

Note: This method is useful if multiple browsers are opened as it creates a unique URL for a browser using random number.



Naina Dhiman is a test engineer with India's leading IT giant Infosys Technologies. She has around 4 years of experience and worked extensively in automation testing using QTP. She is

always enthusiastic in learning new technologies, testing tools and finding bugs. She is User Access Management Domain expert. She believes in work easy and more accurate by revolutionizing the working methods in testing. Besides her work, she loves reading and cooking.



Want to write for Testing Circus?

**It's easy. Read our article
submission guidelines -**

[Click Here](#)



Article Submission Guidelines

Do you have something to share with the testing world? We can make your voice heard to testers.

<http://www.testingcircus.com/article-submission-guidelines>

Article submission guidelines –

- Subject of article can be based on any area of Software Testing. If you want to publish your article on theme based subject please read our announcement of monthly theme published in our site. Article can be submitted without any theme based subject.
- There is no minimum and maximum length of article. If you feel the article is lengthy, please divide the article into logically separated parts so that we can print them in a monthly series.
- Give a meaningful title to the article. If you want a sub-title as well, then add that in a different line.
- Add images/pictures if necessary. If you are using any image/picture which is not yours own work, please include the source. Take care of copyrighted materials. Images need to sent separately with the article.
- Send us the article in MS word (doc/docx) format only. Pdf files are not accepted.
- Write a short write up on the author(s). Usually 7/8 liners in 3rd person descriptive language.
- Include photograph of author(s). Preferred in high resolution .jpeg/.png format. Ideal size would be 50mmX 50mm.
- You can send your article any time. Since we publish every month, your article can be included in any month. There is no such thing called a dead line.
- Send in your article to editor@testingcircus.com with a subject line "Article for Testing Circus – Author Name – Title of the article"
- If you think you can write a column in Testing Circus for at least 6 months, please submit 3 articles in advance. We are open to any idea that may improve the user experience of Testing Circus.
- **(*New)** By submitting article to Testing Circus, you *also* agree to publish the article in our article section and/or in our blog section in the website.



THINKING WHAT TO DO WITH YOUR CAREER?

WE CAN HELP

Tools Journal Testing Corner

Testing Circus & Tools Journal

Testing Circus in exclusive partnership with Tools Journal (<http://toolsjournal.com>) presents the Tools Journal Testing Corner.

1. **Optimize System Integration Testing Efforts**
2. **Solving Data Driven Test Automation Mysteries With Telerik Test Studio**
3. **Towards Test Automation, What & What Not To Automate**

Thank You

"Thank You" to all subscribers who have joined us and have been giving us some good feedback. Most of all Thank you "Testing Circus" for providing us a good platform to shout our views .



About Us:

A start up journal and aspiring social community with an aim to gain and distribute knowledge on software tools and concepts in Testing, Agile, Cloud, Mobile and Enterprise Integration.

<http://www.toolsjournal.com>

With over 500 products listed with quality articles, product owner interviews, we are moving swiftly to launch product editorial/user reviews, community module in next 2 months.

Connect With Us



@toolsjournal



www.toolsjournal.com



www.facebook.com/toolsjournal

Optimize System Integration Testing Efforts

By leveraging system integration testing methods, QA teams can view the performance of in-development software from a user-end perspective.

Quality assurance teams spend a great deal of time testing the individual components of software and ensuring that everything is functioning correctly at a granular level. However, it's always advantageous to step back and take a more holistic view of the product in development. Once a company's latest software suite has been released, the only functionality consumers will care about is how the product performs. To ensure that production is on the right track, QA management should periodically employ system integrating testing to see how the individual software modules operate as a single, functioning unit. This will give development and test teams a better idea how production is progressing and notify department leaders about any drastic performance issues that need to be addressed.

A major tenet of agile development is that by synchronizing QA and development efforts and integrating continual software testing, department administrators can catch software bugs early and correct them quickly. This is an important component of the agile philosophy as software flaws caught later in the development process will be much more embedded in the code and require more labor to ameliorate. The holistic view provided by system integration and black box testing measures can help officials identify software problems earlier in the production cycle.

Software testing expert David Bowman stated that several components should be present in any integration testing effort, including:

- System and volume testing - QA team members should test to ensure that the system functions as

intended and can handle the amount of data it will be tasked with once it has been commercially released.

- Regression testing - A major issue that crops up in software development is the tendency for new applications to conflict with legacy software. Regression testing can examine the functionality of different programs when running in tandem.

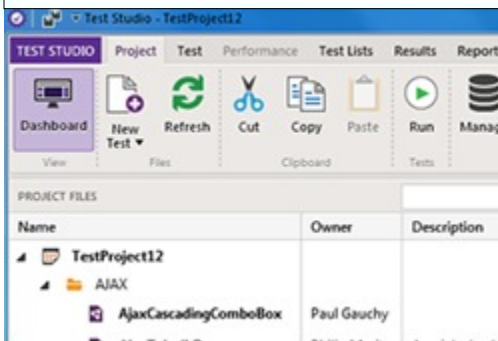
Bowman also noted that any bugs or flaws should be recorded in a bug report and categorized by their severity so development teams know which issues to address first.

Taking the long view on development progress

The University Corporation for Atmospheric Research's Software Engineering Assembly explained that there are different ways QA teams can go about running system integration testing processes, with more complex software requiring a more top-down approach. TechTarget stated that as testers near the end of the production cycle, they will want to increasingly leverage system integration testing to ensure that the software will work properly in a front-facing capacity.

"The goal of systems integration testing is to ensure that the data crossing the boundary between systems is received, stored and used appropriately by the receiving system," the source stated. "Until integration begins, testing of the isolated systems is done on mocked or replayed data and not on "live" data. Integration testing is the final step before customer acceptance testing."

Solving Data Driven Test Automation Mysteries With Telerik Test Studio



A lot of automated functional testing tools support Data Driven Test Automation, let's take a deeper dive into what it is and how does Telerik Test Studio facilitate it, for a stronger grip on the subject.

What is Data Driven Test Automation?

Automated tests can be made more powerful through the use of external data. Data-driven test automation lets us repeat a set of test steps automatically; using a set of pre-defined values both as input and to test the application's output. We can not only test an application more thoroughly this way, we can share the task of data creation which is a

great method for allowing the whole organization take ownership of product quality.

Scenarios Where Data Driven Test Automation is helpful

- Acceptance Test Driven Development (ATDD) where the whole team collaboratively discusses acceptance criteria, with examples, and then distils them into a set of concrete acceptance tests before development begins.
- When the code behind a function is particularly obtuse or scenarios where testing is best done via the application itself.

Test Studio's Data Driven Test Automation

Test Studio can be used for data driven testing. It supports five different data sources: Local Data Source, Excel spreadsheet, XML file, CSV file, SQL database.

You can create an external data driven test by following these steps:

- Add a data source to your test project - select from where to get the data.
- Bind the data source to your test - since you may have multiple data source definitions in your test project, you need to bind your data driven test to a specific data source.
- Attach the data source columns to input values - connect the inputs of your test to specific columns of the data source.
- Attach the data source columns to verifications - connect the expected values of verifications to specific columns of the data source.

An Example for Clearer Vision

In this example, a spread sheet of values to test a sales tax calculation is considered, which the Business Analyst has verified that the formulas being used are correct. The test set includes scenario that may be of particular importance, adding to the whole organization's ownership of quality.

	A	B	C	D
1	Quantity	Price	TaxRate	Tax
2	1	\$ 1.00	5.00	\$ 0.05
3	3	\$ 3.00	3.50	\$ 0.32
4	27	\$ 6.52	7.33	\$ 12.90
5	13	\$ 19.99	7.33	\$ 19.05
6	5	\$ 15.99	5.50	\$ 4.40
7	3	\$ 3.00	3.10	\$ 0.28
8	3	\$ 3.00	3.11	\$ 0.28
9	3	\$ 3.00	3.12	\$ 0.28
10	3	\$ 3.00	3.13	\$ 0.28
11	3	\$ 3.00	3.14	\$ 0.28
12	3	\$ 3.00	3.15	\$ 0.28
13	3	\$ 3.00	3.16	\$ 0.28
14	3	\$ 3.00	3.17	\$ 0.29
15	3	\$ 3.00	3.18	\$ 0.29
16	3	\$ 3.00	3.19	\$ 0.29
17	3	\$ 3.00	3.20	\$ 0.29
18	3	\$ 3.00	3.21	\$ 0.29

Using Test Studio, it's a simple task to create a test for the page with one set of values, then bind that to the spread sheet and let it run automatically. Test studio reads the data, fills in the input fields and checks the output against the expected values. In this particular scenario, users were able to check 150 different sets of values in just a few minutes.

TestResults			
tax calc check list Tests - Test 1 Check tax calculation Iterations			
Fail - 72 data iterations passed out of total 150 executed.			
Iteration	End Time	Passed/Tc	Result
[#1: (Quantity=1)(Price=1)(TaxRate=5)(Tax=0.05)]	5/20/2013 8:40:43 AM	6/6	✓
[#2: (Quantity=3)(Price=3)(TaxRate=3.5)(Tax=0.32)]	5/20/2013 8:40:44 AM	5/6	✗
[#3: (Quantity=27)(Price=6.52)(TaxRate=7.33)(Tax=12.9)]	5/20/2013 8:40:44 AM	6/6	✓
[#4: (Quantity=13)(Price=19.99)(TaxRate=7.33)(Tax=19.05)]	5/20/2013 8:40:45 AM	5/6	✗
[#5: (Quantity=5)(Price=15.99)(TaxRate=5.5)(Tax=4.4)]	5/20/2013 8:40:45 AM	5/6	✗
[#6: (Quantity=3)(Price=3)(TaxRate=3.1)(Tax=0.28)]	5/20/2013 8:40:46 AM	5/6	✗
[#7: (Quantity=3)(Price=3)(TaxRate=3.11)(Tax=0.28)]	5/20/2013 8:40:46 AM	5/6	✗
[#8: (Quantity=3)(Price=3)(TaxRate=3.12)(Tax=0.28)]	5/20/2013 8:40:47 AM	6/6	✓
[#9: (Quantity=3)(Price=3)(TaxRate=3.13)(Tax=0.28)]	5/20/2013 8:40:48 AM	6/6	✓
[#10: (Quantity=3)(Price=3)(TaxRate=3.14)(Tax=0.28)]	5/20/2013 8:40:48 AM	6/6	✓
[#11: (Quantity=3)(Price=3)(TaxRate=3.15)(Tax=0.28)]	5/20/2013 8:40:48 AM	6/6	✓
[#12: (Quantity=3)(Price=3)(TaxRate=3.16)(Tax=0.28)]	5/20/2013 8:40:49 AM	6/6	✓
[#13: (Quantity=3)(Price=3)(TaxRate=3.17)(Tax=0.29)]	5/20/2013 8:40:49 AM	5/6	✗

It can be quickly seen where the problems are — in this case a developer was truncating a decimal value where they should have been rounding. Test studio allowed users to quickly determine the cause, get it resolved, and re-run the tests to everyone's satisfaction.

For a more in-depth look, see video [Data-Driven Test Automation Using Test Studio](#) on Telerik TV, and the code for the demo app used is on github.

Towards Test Automation, What & What Not To Automate

Telerik Test Studio is a Tool for automated functional testing, but the guys at Telerik instead of pushing the tool for automating everything that comes your way, have involved themselves in the process of educating the end users to be discrete in identifying what TO test and what NOT TO test, to make Test Automation truly worthy for the testers, developers and the enterprise which invests money in buying test automation tools.

Here is a post inspired by one of the Telerik's [blog post](#) that will be helpful in defining the boundaries between Manual and Automated Testing.

Using unit tests and test-driven-development techniques; automating integration, functional and user interface tests; and including them at various points in the product's build process are considered to be the "best practices" in any product development organization. These "best practices" are considered to save time, money and increase productivity.

Reducing the number of manual steps, letting computers do the menial tasks makes sense. The code, integration of components, and user interfaces all need to be tested, and where reasonable those tests should be automated. But the key to attain best results out of automation is not to go overboard. Here are some scenarios where Manual Tests would be more efficient.

- User Interface Testing

From the standpoint of user interface testing, not every test should be automated. Layout, colour, and visual clues are best tested by eye. In most cases fully-automating a visual inspection test just isn't reasonable or worthwhile. There are ways to use automation to make your manual process easier and faster. Telerik Test Studio with Fast Forward feature could help you

do just that. As [Kobi Halperin](#) put it on Twitter, "unproductive automation can be an even greater thief than no automation."

- Automating One Off Apps

"One-off" apps built by development team for a specific, short-term event in their projects are seldom used once the project is completed. So time invested in automation of these apps could result in a missed deadline, which simply cannot be tolerated. .

Similarly every company, every development team, every project has its own priorities and decisions need to be made intelligently. Test automation, like any other practice or methodology, isn't "magic sauce" to be mindlessly poured over everything; it should be used where appropriate. [Dale Emery](#) recently wrote that A good automated test will satisfy

- At least one good reason to be a test
- At least one good reason to be automated.

What are your views on the topic do let us know in comments.

Testing Circus

Volume 4 - Edition 9 - September 2013

3rd Anniversary
Special Edition

Interview with
Jay Philips

Your Monthly Magazine
on
Software Testing

- A Fake Tester's Diary
- How to Find A Good Tester
- 40 Definitions of Testing
- Autobiography of Automation
- User Experience Testing
- Is That Testing?

www.TestingCircus.com

To Subscribe
[Click Here](http://www.TestingCircus.com)

Testing Circus

www.testingcircus.com

Still relying on
reading
Testing Circus
from tweets
& facebook
updates?

Subscribe
just with your
email id and
get the
magazine
delivered to
your email
every month,
free!

Testevents for October 2013

Tuesday 01 October
TMap dag 2013, Bussum

Thursday 03 October
Bristol Meetup
NOTiCE Meetup

Tuesday 08 October
Testing & Tools day, Madrid, Spain

Monday 14 October
Ignite Australia, Sydney
Pacific NW SW qlty conference (PNSQ)

Wednesday 16 October
NOSQAA: 7 deadly Sins of Mobile Apps, Cleveland Ohio

Thursday 17 October
London Tester Gathering Workshop
Targeting Quality conference 2013
WOPR 21, San Francisco

Testevents.com

Testevents for October 2013

Monday 21 October

SW Test Professionals Fall 2013, Phoenix

Tuesday 22 October

MBT in the testing Ecosystem, Paris

User Conference on Advanced Automated Testing, Paris

Wednesday 23 October

Nottingham STC Meetup

BraTeste

Thursday 24 October

TestExpo 2013

Sunday 27 October

PotsLightning, Berlin

5th VALID, Venice

Monday 28 October

TesTrek 2013, Toronto, Canada

Agile Testing Days

Tuesday 29 October

QA&test, Bilbao, Spain Testevents.com

Advertise with us.

\$50

only

Testing Circus is even loved by testers who do not read blogs and do not have twitter account. Get connected to the testers workforce across the globe.

We are offering huge discounts for 12 months regular ad booking.

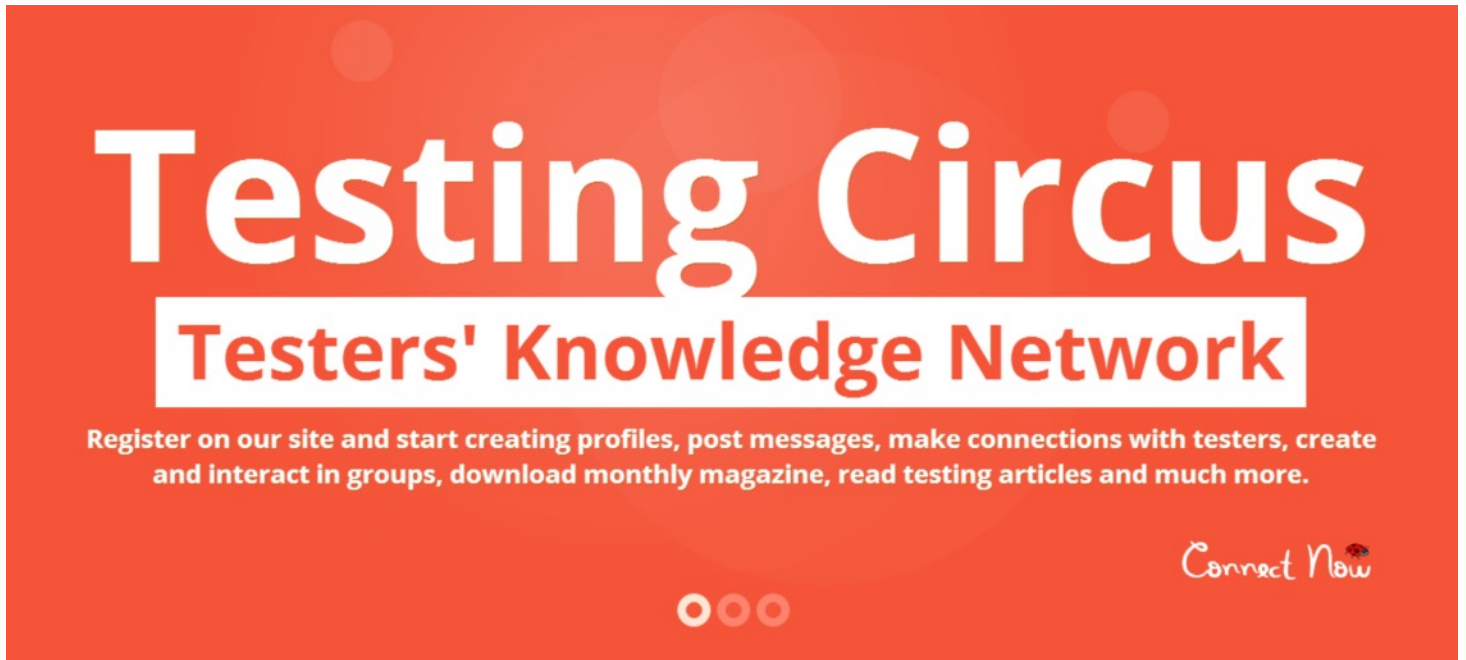


***Advertisement rate starting \$50/per month.

<http://www.testingcircus.com/advertise>

First Look at Testing Circus Community Site

A Community site to connect with other testers, join special interest group, discussing using our forum feature, read articles and download our monthly magazine without any hassle.



Testing Circus
Testers' Knowledge Network

Register on our site and start creating profiles, post messages, make connections with testers, create and interact in groups, download monthly magazine, read testing articles and much more.

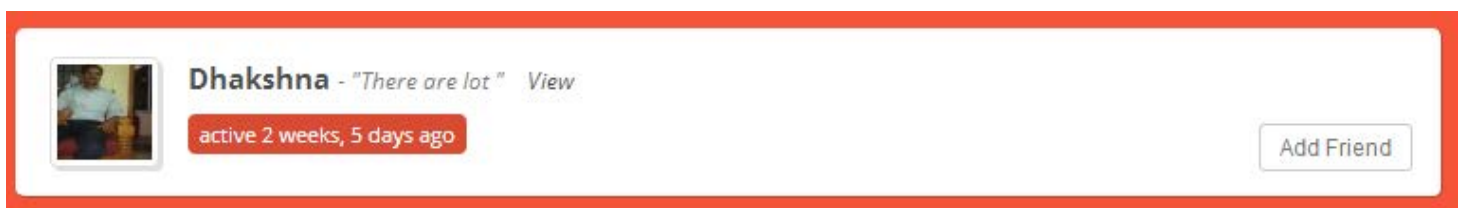
Connect Now


Click on connect now button to register from home page
OR navigate to My Testing Circus > Register



Articles My Testing Circus Write for Us

- Register
- Activate
- Login
- Lost Password



 **Dhakshna** - "There are lot" [View](#)

active 2 weeks, 5 days ago


Add Friend

You can add other members as friend by sending a friend request.

Join some of our Special Interest Groups -


Our Groups

Popular	Active	Alphabetical	Newest
---------	--------	--------------	--------




Performance Testers

Active 3 weeks ago
1 Member




Exploratory Testers

Active 10 minutes ago
1 Member



Hacking Army







Active 6 minutes ago
1 Member



Testing Jobs

Active 1 minute ago
1 Member

Quickly track activities of other members, read blog, join forum discussion -

 ACTIVITY	 BLOG	 FORUMS	 GROUPS	 MEMBERS	 CONTACT
---	---	---	---	--	--

Submit article directly via contact us form -

So what are you waiting for? [Register now](#) and explore a new world of Testing Circus.

Allowed file types .zip | .txt | .pdf | .doc | .docx | .png | .jpg | .bmp | .gif

No file selected.

Are you a human being?