THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

AUGUST 2014 | YEAR 4 ISSUE VII

Jerry Weinberg
Managing in Team Environment

Iain McCowatt
STOP ISO29119

Paul Gerrard
IoE- Architecture and Risks

Sanjay Joshi
Automated Accessibility Testing

Georgios Kogketsof
Road to Test Automation

T Ashok
Load testing is not bombarding the system with concurrent users

Over a Cup of Tea with James Christie


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

The next big thing happening for testers is finally here!

Introducing....

TV for Testers

Your one stop shop for all software testing videos

Alessandra Moreira on "CAST Live" – TI

1 week ago 0 1 20

Karen N Johnson on "CAST Live" – Cele

1 week ago 0 1 19

Standards – promoting quality or restric

1 week ago 0 1 21

Coyote Teaching: A new take on the art

3 weeks ago 0 2 156

How to Talk to a CIO About Software Te

1 month ago 0 1 36

Becoming a Software Testing Expert – J

1 month ago 0 1 48

www.tvfortesters.com

TEA-TIME WITH TESTERS

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

Created and Published by:

Tea-time with Testers.
B2-101, Atlanta, Wakad Road
Pune-411057
Maharashtra, India.

Editorial and Advertising Enquiries:

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

Editorial

About traditions, standards and things like that!

I remember that I had promised to tell you stories from CAST conference when I come back.

Well, I am back now but the story that I am going to tell you today is not directly about CAST but is similar to one amazing thing that happened at CAST this year.

This story is from Mahabharata. Pitamaha Bhishma was very angry with Arjuna because he eloped Subhadra. This was against the tradition because Subhadra's brother Balrama had already promised for Subhadra's marriage to Duryodhana.

Krishna tried to convince Bhishma about how Arjuna was right but Bhishma was adamant. The conversation that then happened between Krishna and Bhishma is what I want to talk about today. I will let you read their conversation first.

***Bhishma**: I am not happy with this marriage because this is against the tradition. A person can trust on another person only when traditions are followed and respected. Without following traditions, a trust cannot be established.*

***Krishna:** (Giggles) Traditions…! Traditions are like mango fruit. In the beginning it tastes bitter. Some days later it becomes sour. People, who like sour taste, eat that fruit with great pleasure. After some days, mango becomes sweet and thus becomes everybody's favorite.*

But, after some more days, it gets rotten. It starts smelling bad. People fall ill who still eat that rotten mango. And as time passes on, that fruit remains of no use. Not even its pit.

I don't have any problem with traditions as such, but when same traditions become reason for one's exploitation, cause more harm than good and become hurdle in the change for the better, then such rotten traditions should be destroyed and new traditions should take birth.

Now you must be wondering, why am I talking about this? I read and re-read this conversation and felt that what Krishna is talking about traditions, is very similar to what standards and best practices are doing to our profession today.

Followed by excellent presentation by James Christie at CAST, some of our colleagues in community have come up with petitions to stop ISO 29119 testing standard. I feel that they are right in what they are fighting for. I would urge you to read this, this, this, this or this post if this whole controversy makes you curious about it.

People who are against this opposition may question, "Who decides if standards are good or bad for our profession? You guys?"

Coincidently, Bhishma had asked the same question to Krishna. I will let you read their remaining conversation.

Bhishma: *Who decides which traditions are rotten? You, Lord Krishna???*

Krishna: *No! Time will decide it, Bhishma. Time! And no one can disobey time's order.*

> *This is the period of solstice for entire Aryavarta. The way Sun changes its path in solstice; whole Aryavarta too is going to change its path. Old traditions will break, old dynasties will get destroyed. New dharma will be established and new era will begin, Bhishma.*
>
> *And a time will come when you will have to decide on which side you want to be. Because, everyone will be caught in this trap of time. People who promote lies will become cause of this transformation and people who walk on path of truth will establish the new era.*
>
> *And this is decided!*

Krishna's answer sums it all. I have signed the petition and I'm done with my part. It's your turn to decide on which side you want to be.

Yours Sincerely,

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

QuickLook

Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

In the School of Testing

T ' Talks

Over a Cup of Tea with James Christie

Family de Tea-time with Testers

QASymphony Names David Keil as CEO

ATLANTA, GA – July 14, 2014—QASymphony (www.qasymphony.com), a leading provider of scalable QA software testing tools for Agile developers, today announced the appointment of David Keil as the company's new Chief Executive Officer, effective immediately.

"We're delighted to have David Keil take over as CEO of QASymphony," said QASymphony co-founder Vu Lam. "David is a seasoned veteran with deep executive leadership experience. We are confident David will apply his strategic vision to scale the company up to the next level of growth and customer acquisition."

"QASymphony has invested heavily over the past several years to create a highly differentiated solution. I am excited to elevate our sales, marketing and partner efforts and significantly expand our reach," said Keil.

"Agile development is spreading rapidly and dramatically impacting how software is tested and measured for quality. QASymphony is uniquely positioned to bring real-time testing intelligence and data to software developers and testers, regardless of their legacy testing platform."

David Keil was previously CEO of Digistrive, Inc. of Atlanta, a provider of e-commerce solutions to large membership based organizations (recently acquired by a large private company). Prior to that, Keil was CEO of Integrated Broadband Services, a software and services company. During his tenure at IBBS, the company doubled in size, completed two strategic acquisitions, and established a market leadership position in the "Tier Two" broadband space. Before joining IBBS, he served as Chief Strategy Officer and later as Senior President and General Manager at ChoicePoint, a NYSE global information services provider (acquired by Lexis Nexis in 2008).

Keil has also held senior executive roles at Novient, and Robinson Humphrey (now part of SunTrust). He holds an MBA from The Wharton School and a BS in Applied Mathematics and Economics from Brown University.

Keil succeeds Vu Lam who has been serving a dual role as both CEO and head of Product Strategy. Lam, a co-founder of QASymphony and KMS Consulting will continue in his role as Chief Product Officer.

About QASymphony

QASymphony's test management and agile testing solutions help teams create better software. With QASymphony's tools businesses can accelerate testing to keep up with the pace of today's development to ensure that productivity gains from Agile development are matched with the oversight, visibility and control required to build quality software. Empowering quality software at companies such as Silverpop, BetterCloud, and Zappos, QASymphony enables teams to communicate and collaborate faster, bringing visibility and control back to the development and testing lifecycle. The company is headquartered in Atlanta, GA.

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

# # #

Press Contact:

Victor Cruz

Principal, MediaPR

vcruz@mediapr.net

Back To Index

DevOps Summit Chicago: DevOps & Continuous Delivery: Successful Adoption

18 September 2014, Chicago
Programme | **Register online**

Adopting a new capability requires a plan that includes people, process, and technology. Although the name DevOps suggests development- and operations-based way of working, DevOps is an enterprise capability that includes stakeholders, business owners, architecture, design, development, quality assurance (QA), operations, security, partners, and suppliers. Excluding any stakeholder will lead to incomplete implementation of DevOps.

The expert practitioners and thought leaders at this DevOps event will help you to consider the challenges within your organization and develop your business case and build the foundation towards getting significant return on investment.

The thought leaders are – Dave Snowden, Cognitive Edge; Sam Eaton, Yelp; Stephany Bellomo, SEI; Sanjeev Sharma, IBM; Andi Mann, CA Technologies; Alex Papadimoulis, Inedo; Jeff Downs, Tasktop Technologies; Jeremy Alons, Spot Trading introduced by SaltStack.

Exploring Software Testing: Strategies & Innovation

18 September 2014, Chicago
Programme | **Register online**

The conference features a highly interactive programme- case studies and traditional presentations interspersed with panel sessions. Discussions with the thought leaders and practitioners are based around testing approaches that are at the forefront of innovation and those that can be adopted to deal with advice on the recent cutting edge software technologies.

Among the prominent speakers who will help you to develop and formulate a strategy which is applicable for your own requirements are: Dave Snowden, Cognitive Edge; Donald Firesmith, SEI; Dorothy Graham, Software Testing Consultant; Iris Trout, TD Bank; Stan Adams, Prolifics.

Discounts:

SPECIAL CONFERENCE PRICE: $299 per delegate
HALF-DAY/ONE-DAY WORKSHOP BOOKED WITH CONFERENCE: conference special price: $299 + $99 (half day workshop)/ $199 (one-day workshop) per delegate.

How to book:

Code for claiming special rates: quote "**Tea-timeSPECIAL**" in the comments field on the booking form to apply special rate.

A million dollar smile ?

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

*Adverts starting from $100 USD | *Conditions Apply

Photo credits: Amir Eltahan

James Christie is quite popular and respected name in community of thinking testers. He is a self-employed testing consultant and has 30 years experience in IT. He has worked as a test manager, IT auditor, information security manager, project manager, business analyst and developer. He is particularly interested in the links between testing, governance and compliance. He is a member of ISACA, the Information Systems Audit and Control Association. James spent 14 years working for a large UK insurance company and then 9 years with IBM working with large clients in the UK and Finland.

James has been self-employed for the last 7 years. I got to spend quality time with James when I met him at CAST conference this year. His amazing talk around testing standard (which he presented at CAST) has sparked a candid movement in community. Apart from his contribution to the community, what has earned my respect for James is his down to earth nature. His thoughts are powerful but he is equally humble.

I feel privileged to know and meet James in person. Find out what all we discussed at CAST in this exclusive interview.

- Lalitkumar Bhamare

Over a Cup of Tea with James Christie

Thank you for talking with us today, James. You have seen computer software field for over 30 years. How do you feel when you look back in past?

I'm torn between gratitude that I've had the opportunity to do many different roles, and regret that I didn't settle on one. My variety of experience has given me a better overall understanding of how IT fits into the business than I would have had if I had stayed in one specialisation. However, I do feel that moving around has stopped me making a mark in one area. If I had concentrated on one profession then I might have made more of a difference.

I doubt if I'll ever resolve that. So I just shrug and move on. When I look back I think the time in my career I'm most proud of was when I worked as a Y2K test manager. The problem in the area I was working on was much bigger and more complex than either my employer or the client had realised. I had to persuade them how difficult the job was going to be, then come up with a feasible test strategy that would provide the most reassurance possible in the time available, then make the strategy work. We finished the planned work in time, just in time! I think it was the only time I felt as a test manager that I'd done as good a job as I could have done.

What do you think has changed about testing recently; that you think should have changed long time back?

I think it's great that the Context Driven School has become more mainstream. Opponents can no longer dismiss it as being irrelevant to serious corporate work. I think that's a vital development. It's a pity it didn't happen earlier. If it had done then ISO 29119 might have been a non-starter. People would have seen it for the irrelevant and damaging dead end for testing that it is.

You have worked within organisations as well as an independent test consultant. What difference does it make on one's perception about quality in either of the roles?

When I've worked as an employee, or as a contract test manager, I often felt that quality was just desirable, rather than essential. Organisations often pay lip service to quality. The bottom line is that there is a process to follow. That becomes the goal, rather than achieving something worthwhile. It can be hard to change that. I think it's easier for an external consultant to tell uncomfortable truths like that. Mind you, it doesn't necessarily go down well. Some consultants realise that and tell clients what the client wants to hear. I think that lacks integrity. You have to be prepared to be thrown out for telling it like it is. Otherwise you're taking money under false pretences. It's not just individual consultants who do that. Big ones are also guilty, and possibly more so.

Which job is more challenging and why?

There is no simple answer to that. I think most people would find the independent consultant role much more stressful and difficult. It is also difficult to find work. However, I couldn't go back to working in a way that I believe was wrong and ineffective. I would find that too difficult. When I was younger I was more impressionable and I was more prepared to believe that big companies knew what they were doing and that I needed to adapt to them. The thought of being independent was far too challenging, and I needed lots of experience before I had the confidence to become self-employed.

What are some most common mistakes that happen in test management? What would you advise to avoid them?

.... CONTINUED ON PAGE 61

Managing in a Team Environment

2.5 Controlling

Managers make two kinds of mistakes concerning the amount of control they exert over internal team affairs: too much or too little. Pattern 1 managers tend to intervene too little, placating the team. Pattern 2 managers tend to overcorrect by intervening too much and blaming the team, often in the form of claiming to be better able to do the job themselves:

The viewpoint of business executives who usurp the function of lesser managers only to discover that in the real world no improvement results has been well summarized by the baseball manager who yanked his centerfielder after he dropped three straight fly balls. Having decided personally to take the place of the errant fielder, the manager suffered the ignominy of himself dropping what proved to be the game-winning pop fly. Returning to the dugout and the penetrating stares of his players, the dismayed manager explained, "He has that position so fouled up that now no one can play it."

Both placating and blaming usually stem from feelings of personal inadequacy—that is, from the manager's internal needs rather than the team's.

How to tell if the team needs an intervention

For effective steering (as in a Pattern 3 organization), managers' interventions or non-interventions into team affairs must be done on the basis of what the team is doing, not what the manager is feeling. Ironically, you as the manager can tell a great deal about whether or not the team members are working well as a team by observing their emotional state. Here are examples of what you can observe in a well-functioning team, followed by parenthesized examples of what you may hear to indicate a team malfunctioning:

1. Well-functioning teams make sure that everybody participates in decisions. ("Nobody asked me if I wanted to work overtime to get the release finished.")

2. Members stay in touch with one another, so they have the information needed to make their decisions. Even if there are no specific decisions to make, the contact must be maintained. "No social group—whether a family, a work group, or a school group—can survive without constant informal contact among its members." ("I never heard that the interface was changed." "I haven't seen Phil for a couple of weeks.")

3. All team members feel that they each have a chance to contribute. ("They never listen to my ideas.")

4. Team members are united. ("It's not my fault that Wally didn't follow the recommended process."

5. Closely related to the necessity for unity is the team's feeling good about "what we did." (Any statements opposing "me" or "they" to "we" are indicative of less than optimal team functioning. For example, "Their work isn't up to the standard of my work.")

6. The team has fun. This one seems exceptionally difficult for certain managers to accept. Apparently they believe that "fun" and "work" are opposites, and that you shouldn't be paid for having fun. Not all fun teams are productive, but all productive teams enjoy themselves. (Some of the team members may share the fun/work dichotomy, so you have to be careful how you interpret complaints about how hard the work is. "We're killing ourselves" could be a complaint about not having fun, or it could be bragging about how proud they are of how well they're doing.)

7. Team members rely on each others' individual strengths, and all do what they can that's best for the team. ("I did that myself because Wanda wasn't likely to get it right, even though it was assigned to her." "I'm a data base expert, and I'm certainly not going to spend my time preparing test cases for Jack's module.")

8. When members speak, they take care to be sure that everyone understands. ("I haven't got time to explain that to Sarah.")

9. Team members each feel strong and useful, but not out of proportion to their competence as a team. ("Well, my part isn't having performance problems." "Go on without me; I just can't keep up with Alex and Demma.")

10. Members show how they are truly feeling. ("I'm not going to let Harry know how angry I am he did that.")

Intervening for the team's benefit

Of course, you don't use a single incident to trigger an intervention. Instead, you use it to trigger a state of alertness for other signs. If other incidents reinforce your interpretation, you can reframe them into opportunities for team building and team problem solving. For instance, suppose the last piece of evidence of team trouble is when Marilynne tells you, "I'm not going to let Harry know how angry I am he did that." You could get yourself in the middle by talking to Harry yourself, or you could try to convince Marilynne to talk to Harry.

But here's an approach more congruent with your desire to build the team. You say to Marilynne, "When I hear you say that you're angry with Harry and you aren't going to let him know, I get worried about the way your team is functioning. I think an effective team needs to be able to deal with its members' feelings, no matter how strong or negative they may be. I'm going to call a team meeting, raise this issue, and sit back and watch you resolve it."

Marilynne may protest, but probably without great conviction, for why do you think she brought this up with you in the first place? Of course, she may have thought you were going to tell Harry for her, but this way you demonstrate to her that you expect the team to solve problems like this for itself—perhaps with a bit of help from you, early on.

Envy of the well-functioning team

Seeing the list of attributes of a well-functioning team, we can understand how easy it is for managers to envy their best teams. The cure for that envy is to reframe it.

At the first pangs of envy, you would do well to reframe your model as described by my colleague Mark Weisz:

Any managers job can be terribly lonely at times, but hopefully it isn't all that grim. The greatest joys I experienced as a manager (and there were many) were when I brought people together to work successfully as a team. There is something especially gratifying in creating a sound work team and, until the next crisis at least, you can relax a bit and watch it hum like a finely tuned engine. There's no better feeling. When the next crisis comes, as it inevitably will, you've built in a reserve of resilience to handle it successfully. That team is your insurance policy. Management is a noble profession. Like any job, it has its frustrations and pain, but also the capacity for great achievement.

If you can cultivate a sense of humor (and perhaps Stoicism) to get you through the tough times, you'll probably find you have a lot more good ones.

Rewards that don't reward

Managers who are uncertain about how to interact with a team often try to win their friendship by giving them rewards. This is not as foolproof a strategy as you might suppose. Brooke, a member of a client team, told me how their manager had "rewarded" them for heroic work to salvage a dying project:

We had worked extensive unpaid overtime for about two months to get the release out the door. The day after the release, we each found on our desk a one-dollar gift certificate from the local coffee shop. There wasn't even a note with it, but someone asked and he explained it was his way of thanking us. Several of us were so enraged we returned them in his mail slot. The calmer ones just ignored it. I personally would have preferred he spend the dollar on a card, which he could have signed with his own name and handed to me personally. If he wasn't capable of that much human interaction, I would have preferred he take his dollar and ... (I have deleted what Brooke said she wanted him to do with the dollar.)

Rewarding congruently

If you fear that you, too, may be as socially inept as this manager, here's a suggestion from Georgia, one of my students, about how she handled this managerial situation congruently:

When I got back from class, I called the team together and said, "In addition to all the other things you've accomplished, you've really saved my skin. I truly want to find some way to show my appreciation to the whole team, but I'm not very sensitive about these things. I'm afraid that I won't know how to express myself in a way that you felt was really appropriate. Would you help me find some way?" Then I trusted them.

Georgia was amazed at what this congruent expression of her position could accomplish. You will be, too. What about punishing them? Can this be done congruently, too? Frankly, I don't think there's any way to punish congruently, and that's because it's not your job to punish people. If you have delegated a properly formed task in an appropriate way, and if you have exercised appropriate control, then if the team fails, the members will know without doubt that they have failed, and that will be more than enough punishment. They will also know—and so will you— that in this failure you are part of the team, so who are you to punish them?

Instead of punishing, you'll be better off arranging for opportunities to learn.

The middle manager's job is to "grow people"—not to build a file of lifeless documents, or to spend half of each day in boring meetings. The middle manager role means walking around with an open mind toward listening and helping, and taking time to talk things over. It means being concerned about individual welfare and fostering the full potential of each person.

One congruent way to do this is to sit down with them and say, "I feel bad because we blew this one. Obviously, there's something I didn't know about how to accomplish this job, and I want to learn what it is so I can do better next time. Can you help me with this, and is there anything you want to learn that I may be able to help with?"

To be continued in next issue...

Back To Index

Biography

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

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

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

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

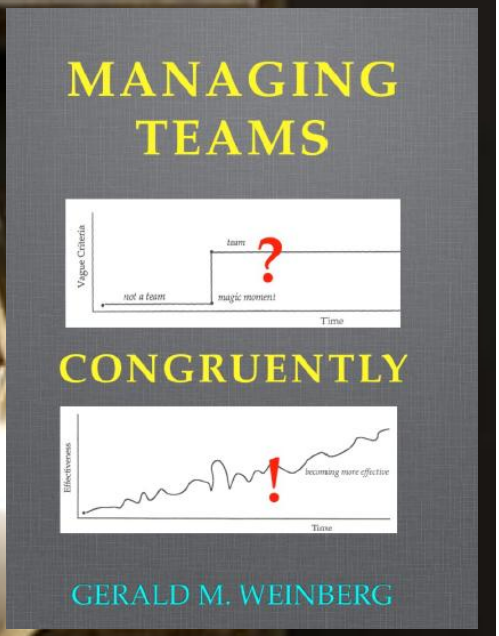
Gerald can be reached at hardpretzel@earthlink.net or on twitter @JerryWeinberg

To be effective, team managers must act congruently. These managers must not only understand the concepts of good software engineering and effective teamwork, but also translate them into their own practices. Effective managers need to know what to do, say what they will do, and act accordingly. Their thoughts and feelings need to match their words and behaviors.

And how should they do that? Jerry has shared this secret in his **MANAGING TEAMS CONGRUENTLY** book.

Its sample can be read online here.

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



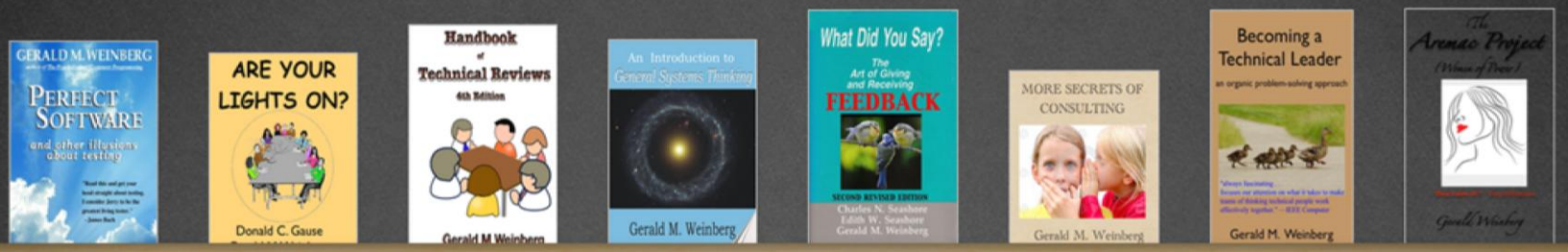
TTWT Rating: ★★★★★

The Bundle of Bliss

Buy Jerry Weinberg's all testing related books in one bundle and at unbelievable price!

The Tester's Library

Sold separately, these books have a minimum price of **$83.92** *and a suggested price of* **$83.92**...



The suggested bundle price is **$49.99**, and the minimum bundle price is...

$49.99!

Buy the bundle now!

The Tester's Library consists of eight five-star books that every software tester should read and re-read. As bound books, this collection would cost over $200. Even as e-books, their price would exceed $80, but in this bundle, their cost is only $49.99.

The 8 books are as follows:

- Perfect Software
- Are Your Lights On?
- Handbook of Technical Reviews (4th ed.)
- An Introduction to General Systems Thinking
- What Did You Say? The Art of Giving and Receiving Feedback
- More Secrets of Consulting
- Becoming a Technical Leader
- The Aremac Project

Know more about this bundle

TEA-TIME WITH SMARTBEAR

About this column…

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

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

On the Reusability of Test Scripts

- by **Paul Bruce**

Automating your testing efforts is paramount to agile and expedient software delivery. However, the implementation details of automating tests diverge very quickly down paths defined by what level of testing you're performing. Developers have unit tests in IDEs, QA testers have their own tools and scripts, and operations folks have monitoring – all to make visible what is either broken or performing poorly.

This is not wrong. Your testing strategy must embrace technical diversity.

We do the best we can, but no one is an expert at everything. Each of the more traditional levels of testing (i.e. unit/integration/interface/system) exposes issues with different aspects of how software performs within certain periods of the delivery cycle or under certain conditions. Managing test scripts in and of itself can be a tough gig. Feature delta causing script breakage, code coverage and analysis, compositionality of tests, and scalability of scripts all impact how much our automated tests actually simplify our day to day software quality efforts.

So we'd expect that the implementation details of each testing level differ from each other, right?

Alas, even experienced testers often fall prey to the "this should be simpler" notion, which might be a great long-term direction, but is counter to the very nature of why we do testing to begin with. The statement usually goes:

"Why can't I use this [unit/functional] test as a [load/performance] test?

Shouldn't it be as simple as that?"

We test things because that which is complicated (like software) needs to be checked thoroughly so that it remains as simple as possible. While simplicity in a user's experience or in management process is a good goal, all notions of simplicity should be checked at the door when a tester puts their testing hat and gloves on.

In other words, simplicity is a perspective on outcome, not a technical approach to determining quality.

Take for instance the use of functional, interface-based web tests (like those created in TestComplete, Selenium, etc.) to make sure a feature still works before release. Interface tests by their very nature infer the need to have interactive browser or window session, where concepts like "click" and "type" apply. In contrast, a load test is typically a representative set of traffic between computers, having no concept of "click", just a representation of what happened due to the click. The overhead of "interface" as part of the test dramatically increases the cost of each instance of the test you run.

Functional testing also often requires validation based on data retrieved from a back-end system, for proof that the app completed a logical process *accurately*. However, in a load test, the simple act of retrieving a database value for comparison to the app's results introduces both additional chatter (side traffic to your QA data, etc.) and increased latency (query plus comparison times).

I can't help at this time but mention how extremely unpleasant and how much of a dark art it is to quantify the performance impact of custom scripts, since they themselves take time to execute and are hard to separate from other performance metrics. As a 15+ year developer, I know that when you give people the capability to do something, they *will* do that thing. The more custom script you use to run a test, the more you run the risk of tying your own shoe laces together. This is why the testing industry typically divides functional testing from load testing, in strategy and in tool sets.

There are very good reasons why there is no "one test to rule them all", and vendors who try either hit scalability problems or provide inaccurate results. Mitigating cost and scalability issues by having separate scripts for different levels of testing is reasonable in comparison to the alternative: unwieldy or incomprehensible scripts and test results. We at SmartBear know the minefield that is unified testing and have tools that can safely navigate you to success.

We look forward to hearing about your strategies and testing efforts.

An Experience Report by a First Time CAST Attendee

I am glad to have made it to the 9th annual Conference by the **Association of Software Testing** organization. I witnessed the on-goings first hand and I have here made an attempt to present the learning to you - My CAST journey began at the Kimmel Center in the picturesque city of New York.


Kimmel Center, NYU – Venue for CAST 2014

Volunteers meet on August 10th, Sunday

The volunteers and the board of directors of the Association of Software Testing (AST) met on Sunday evening to discuss about the volunteering work. The proceedings were an after effect of this meeting.

Thanks Anna for leading the way.

Participant names:

Volunteers

Anna Royzman , Keith Klain, Richard Robinson, Smita Mishra, Paul Holland, Bernie Berger, Paul Holland, Mike Lyles and **Helena Jeret Mae**

Registration Desk

The busy registration desk catered to the registrants by providing them with adequate information and directions from day 1 to day 3. Thanks to **Pete Walen**, **Dee Ann Pizzica** and **Markus Gartner** who helped set up the registration desk.

Registrants queuing on the day - 1

Keynotes at CAST2014

- James Marcus Bach
- Trish Khoo
- Carol Strohecker
- Ben Simo
- Matt Heusser

James Bach - Testcases are not testing - Towards a performance culture

If you have read about and followed the context driven testing community and have been an active member of the context driven testing talks online and offline, then you would already know that test cases are not testing.

James was present at CAST2014 emphasizing the need to not consider test cases as testing, providing hilarious analogies of the crab and the booby-trap to testing and checking.

My notes from James's keynote:

I learned from James about how to be a strong advocate of what you believe in despite the oppositions and arguments.

Online mentoring can help in sharing tacit knowledge.

James delivering the keynote

Many mentors have taken it upon themselves to illuminate the software testing world and eliminate the darkness that exists in non-sensical definitions, terms and processes. **James Marcus Bach**, **Michael Bolton**, **Anne Marie Charrett, Huib Schoots** are a few among the many other testers who coach online (via Skype) and I was glad to meet them all in person.

Douglas Hoffman and **James Bach** continued to debate on the method of testing, even after the *open season. And amen to this: *Arguments are common among people with passion for the craft.*

***Open Season** is when the audience gets to ask questions to a speaker by using one of the K-cards which is provided to all the participants.

The facilitators **Paul Holland** and **Richard Robinson** narrated to a room full of audience about the usage of K-Cards. Paul and Richard were great at facilitating

throughout the event and kept the audience educated and entertained. Kudos to both of them!

Trish Khoo - Scaling up with Embedded Testing

Trish presented on her cross-over from software development to software testing. And an experiential report gathered by her talks with various testers / test teams across the industry and the need / lack of developers in testing and vice-versa. As the abstract of her talk conveyed: Trish shed light upon testers and non-testers involvement in shortening the feedback loop between creation and verification of any product. An excellent contribution by Trish towards understanding the delay in responses across teams and she did put forth her point in a convincing way.

Carol Strohecker - STEM to STEAM Advocacy to Curricula

Carol, from the Rhode Island School of Design (RISD) presented about her experiments with design in everyday life. She stuck to the theme of the conference. I agree that design though sounds too artsy. References to Picasso and innovation by design put the science back into the theme of CAST "The art and science of testing".

Ben Simo - There Was Not a Breach; There Was a Blog

Focus of Ben's keynote was Security Testing. The talk engaged the audiences and provided guidelines for securing an online web application and about using HTML Injection, browser add-ons like Developer Toolbar to capture secure information leaks. It did seem like a lot of the issues in healthcare.gov could have been fixed if they tested the application also from a web security perspective. The testing community and the testers sat up and took notice of the security aspect of testing.

Matt Heusser - Software Testing State of the Practice (And Art! And Science!)

Highlights from Matt's talk included:

State and longevity of the software testing practice

Teaching testing

Schools of testing

Some of the testers echoed to what was being talked about:

Testers are empowered to do anything and conveyed the message "Just Do It".

There is no difference between Development and Testing teams or rather we are all on the same level. And I feel that level is → Everyone in the team being responsible for delivering a quality product.

I'd agree with what **Jean Ann Harrison** and later Matt echoed - *Weekend testing requires less commitment, one Saturday in a month and this is where you also get to learn from many other testers across the globe.

***Weekend testing** - online testing sessions where testers are taught skill based testing.

The essence being "Don't just rely on your company trainings. Make your own opportunities - take help from the mentors and learn from each other".

Matt wrapped up his talk with these three words → Honesty, capability and reach as the futuristic mantra for the software testing world.

Live and Recorded Keynotes and tracks

Thanks to Benjamin and Paul Yaroch for live-streaming CAST and Dee Ann Pizzica for co-hosting "CAST Live" with Ben this year. All recorded sessions and interviews are *now available online.*

How did CAST help my learning?

I need an introduction to the subject / sciences.

I need for myself to gather, interact and interpret the knowledge sources.

Be actively present to know what is in it for me.

Then will I be equipped to either advocate or think is it for me or not.

And CAST did help me to learn just this.

This conference indeed was a fast track ticket for me to help learn how I learn.

How I learnt at this conference?

Absorbed as much information as possible generated at the conference.

For the rest of the learning, I made notes, talked to the people to later re-collect from, got on to the *social sharing mediums to learn about the references I noted down.

*Quantifiable amounts of information is shared on the social sharing mediums, like on twitter you can follow the leads of CAST with the hashtag #CAST2014.

And that's not it. There is a lot more to share from this amazing conference. Watch out for my further write up in next issue of Tea-time with Testers.

- Jyothi Rangaiah


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

Following an excellent presentation by James Christie at CAST2014, I participated in drafting a petition calling on the International Organization for Standardization (ISO) to withdraw those parts of the ISO 29119 standard for software testing that have been issued to date, and to suspend production of the remaining parts.

You can find the petition here: http://www.ipetitions.com/petition/stop29119, and this is why you should sign it.

A Warning from History

In 1979, with support from the Thatcher government, the British Standards Institution (BSI) first published the BS 5750 series of standards. These made provision for organizations to become "certified" and display a mark of registration, supposedly a sign of quality. By 1987, the British Government had convinced the International Organization for Standardization (ISO) to adopt BS 5750 as the basis for an international set of standards: ISO 9000.

Adoption was rapid. In the UK this was driven by Department of Trade and Industry (DTI) grants to firms who chose to register, regulation (e.g. the 1993 adoption of ISO 9001 conformance as a requirement of Oftel's Metering and Billing scheme), and market coercion: the threat that large purchasers would only source goods and service from suppliers who were registered. Similar patterns

emerged internationally, with many organizations being convinced that EU adoption of the standard meant that registration was a cost of doing business in Europe. By 2009 over a million firms were registered worldwide: over a million firms supporting an ecosystem of consultants, training providers, and assessors.

Nothing so dramatic has yet been seen in the world of software testing. When compared to the uptake of ISO 9000, IEEE 829, BS 7925 and their ilk seem to have been largely ignored. No wonder many testers seem to view the new ISO 29119 series of software testing standards with apathy.

Yet ISO 29119 could be the ISO 9000 of software testing. Whilst other testing standards were relatively limited in scope (documentation, component testing etc.) 29119's stated aim is to "define [a] set of standards…that can be used by any organization when performing any form of software testing" (ISO/IEC/IEEE 29119-1:2013, Introduction). Any form of testing, in any organization, in any context. That means YOU.

Further, ISO 29119 is designed to support conformance and registration: conformance may be claimed to parts 2, 3 and 4, ISO 29119-2 describes how full or partial conformance with its processes might be achieved, the website of the ISO working group responsible for the standards describes how ISO/IEC 33063 would be used to assess test processes against ISO 29119-2. ISO 29119 has all the makings of a registration regime that would extend to any testing, anywhere. The bonfire is set; all that is needed is a spark to light it.

In the late 70's, such a spark came in the form of Thatcher's desire to repeat the "Japanese miracle" and reinvent British Management. What might ignite ISO 29119? Take banking, with its explosion in regulation following the last crash. Now imagine an event, a software failure, a Flash Crash or Knight Capital writ large, something that causes significant economic damage. Or take an airline, with hundreds of souls aboard. Or drug production. Or any of dozens of examples: software is everywhere. Now imagine the outcry: "*No Senator, despite the existence of internationally agreed standards, we did not apply those standards to our testing*"[1].

It's not terribly hard to imagine, indeed it may already be happening: some vendors are calling for the adoption of standards as a result of the Healthcare.gov fiasco[2]. Love it or loathe it, the one thing you cannot afford to do is ignore ISO29119.

A Flawed Approach

Standards in manufacturing make sense: the variability between two different widgets of the same type should be minimal, so acting in the same way each time a widget is produced is desirable. This does not apply to services, where demand is highly variable, or indeed in software, where every instance of demand is unique.

Attempting to act in a standardized manner in the face of variable demand is an act of insanity: it's akin to being asked to solve a number of different problems yet merrily reciting the same answer over and over. Sometimes you'll be right, sometimes wrong, sometimes you'll score a partial hit. In this way, applying the processes and techniques of ISO 29119 will result in effort being expended on activities that do nothing to aid the cause of testing.

And in testing, that's a major problem. When we test, we do so with a purpose: to discover and share information related to the quality. Any activity, any effort that doesn't contribute to doing so is waste. As "complete" testing is impossible, all testing is a sample. Any such waste results in a reduction in sample size, it equates to opportunity cost: an opportunity lost to perform certain tests. For a project constrained by quality, this translates into increased time and cost. For a project constrained by time or money, this translates into a reduction in the information available to stakeholders, and a corresponding increase in risks to quality.

The 29119 crowd might tell you that the new standard takes this into account, that it encourages you to tailor your application of the standard to each project, that it is sufficiently comprehensive that you need only select the processes and techniques that apply. This is the Swiss Army Knife fallacy: if you have one you'll never need another tool. One of the problems with a Swiss Army knife is that it's not much use if you need a pneumatic drill, or an ocean liner.

Training testers to use a standard in this way has a tendency of framing their thinking. Rather than trying to solve testing problems, they instead seek to choose from a set of ready-made solutions that may or may not fit. I once conducted a highly informal experiment with two groups of students. The first group was trained in a set of formal test design techniques. The second received a short briefing on some general testing principles and the use of the heuristic test strategy model[3]. Both were then tasked with creating a set of test ideas for the same product. The difference between the ideas generated by the two groups was stark. The first group came up with a predictable set of equivalence classes etc., whilst the second group came up with a rich and varied set of ideas. When released, and if widely adopted, part 4 (on test techniques) will give rise to a generation of testers locked firmly inside the 29119 box, without the ability or freedom to solve the problems they need to solve.

And that's not the worst of it. For my sins, I spent a number of years as an ISO9000 auditor. It seemed like a great idea at the time: understand the system, monitor the system, improve the system. Gradually, I realized this wasn't the reality. People were documenting the system, documenting their reviews, documenting their responses to audit findings, and doing very little by way of improving the operation of the business. What the hell was going on? Goal displacement, that's what. We'd created a machine geared towards complying with the standard, demonstrating conformance to the satisfaction of an assessor, and maintaining our registration once obtained. Somewhere along the line, the goal of improving our business had been forgotten. This phenomena isn't limited to organizations that seek compliance with external standards. Not so very long ago, I watched an organization doing much the same whilst attempting to standardize their testing processes. Significant effort was directed to completing templates for test documentation, reporting metrics and self-assessing vs. the internal standard – all with no regard for the relevance or value of doing so for the projects that testing was meant to be serving.

Waste, waste, and more waste.

So when standards proponents tell you that following ISO 29119 will improve the efficiency or effectiveness of your processes, call them out: far from making testing more efficient or effective, conformance will have the opposite effect.

No Consensus

The text of ISO 29119 claims that it is "an internationally-agreed set of standards for software testing". This agreement is meant to be the product of consensus, defined by ISO as "general agreement, characterized by the absence of sustained opposition to substantial issues by any important part of the concerned interests and by a process that involves seeking to take into account the views of all parties concerned and to reconcile any conflicting arguments" (ISO/IEC Guide 2:2004).

There is no such consensus. Instead there is a small group of members of a working group who claim to represent you. Meanwhile, hundreds of testers are calling for the withdrawal of ISO 29119.

There is no consensus; there will be sustained opposition.

Join the opposition: http://www.ipetitions.com/petition/stop29119

NOTES

[1] For more on this theme, read "Uncle Bob" Martin's After the Disaster

[2] My thanks to James Christie for drawing attention to this during his CAST 2014 presentation

[3] http://www.satisfice.com/tools/htsm.pdf

Iain McCowatt is one of the founders of the International Society for Software Testing.

His day job is as a director in a bank, helping the stakeholders of large enterprise IT programmes to solve complex testing problems and gain insight into the quality of their software.

Visit his blog - http://exploringuncertainty.com/blog/

You may want to read some other interesting posts around ISO 29119:

- Fiona Charles – Why I oppose adoption of ISO 29119
- Keith Klain – The Petition to Stop ISO 29119
- Karen Johnson – My Thoughts on Testing Certifications
- James Bach – How Not to Standardize Testing
- Michael Bolton – Rising Against the Rent Seekers

8 Tips on How to Best Manage Distributed Teams

- by David Keil

More and more organizations are using distributed testing teams yet many are hard pressed to know how to manage teams that may be separated by thousands of miles away or on another continent. Too often, companies think about offshore product development only as a means to cutting costs and accelerating time to market. Such an approach can produce poor results with your projects if you don't take into account certain issues like cultural barriers, metrics and workflows.

The smart strategy is to focus on the needs and skills of the team and to implement procedures that will help manage them better. Your standard bottom-line goals -- cutting costs and accelerating time to market -- will take care of themselves if you can follow these 8 tips for managing distributed teams.

1. Know Your Team

It's important to address people correctly. People respond well when somebody knows how to spell and pronounce their name. To infuse a sense of team spirit, attach a face to the person who is communicating via email, phone or IM. Have a kick-off meeting for a project team, where each person introduces themselves with some basic personal information such as a hobby or interest, to help establish a 'human-side' to the offshore team. While meeting in person might not be a possibility, the liberal use of video conferencing can really improve the relationship with remote teams.

2. Eliminate Language Barriers

Working with remote teams on different continents/countries requires strong communication. It's critical for the key contacts between teams to familiarize themselves with the pronunciation and

grammatical styles of each other, while also ensuring that your team members obtains International English certifications, such as TOEFL. In addition, when possible, I recommend that the team leaders from the vendor side and client side visit each other's offices to understand the culture of the companies and to instill a sense of partnership.

3. Clearly Define Roles and Responsibilities

To ensure that all members of the project team understand their expected tasks, it is vital to establish clear roles and corresponding responsibilities early on. Create a collaborative, pyramid-style team structure for the offshore team, so that the key people responsible for daily communication with the rest of the project team are identified, along with an obvious reporting hierarchy.

4. Create Strong Communications

The distance between remote teams working on the same project can be eliminated by use of communication tools such as IMs, VOIP-based phone calls, online meeting tools and other electronic mediums. However, communication is only as good as the information provided. All relevant documentation and domain knowledge must be continuously shared with all team members.

5. Track Detailed Onshore/Offshore Planning and Tasks

To ensure that your teams understand the assigned tasks and are progressing as expected, it is very important to have a milestone-driven WBS (Work Breakdown Structure) that identifies the effort associated with each task, and identifies the timeline of when the tasks need to be completed. This level of detail ensures that everyone is clear on their assignments, and that any delays (and corresponding affects) can be identified as early as possible, so that a mitigation plan can be implemented if need be. Team progress on assignments and activities should be followed up with daily, weekly and monthly reporting, leveraging defined templates where possible.

6. Establish Process Workflows

A clearly defined testing activity workflow should be established and reviewed with the offshore team, to ensure that all team activities are synchronized. Documenting a process workflow allows the team to streamline, validate and verify expected activities that could impact the project delivery timelines.

7. Don't Forget Metrics

Offshore team deliverables and activities should be gauged with quantifiable metrics such as timeline delays, actual-to-estimated effort, QA-to-production defect ratios, and so on. Metrics should be established under at least three categories: timelines, productivity, and quality. These metrics should be established at the beginning of the project, and tracked either by release or on a monthly basis.

8. Establish Roadmap for Refinements

To ensure that the offshore team continues to increase its value to the project team, it is important to leverage the daily, weekly and monthly reports/metrics to identify areas of improvement/streamlining for testing processes and test strategies. These refinements should be prioritized and scheduled for implementation by the team, and tracked for progress along with other project deliverables.

In summary

Establishing and managing distributed teams is a complex challenge. Cultural differences, time zone variations and lack of face-to-face time contribute to the intricate nature of this model. In addition, some companies focus too much on cost-cutting objectives. To successfully manage distributed testing teams, a company needs to focus on people, processes and communications.

David Keil is CEO of QASymphony, a leading provider of test management platforms for agile development teams. He previously served as CEO at Digistrive and at Integrated Broadband Services.

He holds an MBA from The Wharton School and a BS in Applied Mathematics and Economics from Brown University.

Back To Index

Internet of Everything – Architecture and Risks

by Paul Gerrard

In the first article of this series, I set the scene of the wonderful future world of the Internet of Everything. Right now, it is a very confusing state of affairs, but clearly, an awful lot of effort and money is being invested in defining the standards and building business opportunities from the promise of the new Internet world order. Standards are on the way, but today, most applications are what you might call bleeding edge, speculative or exploratory. It looks like standards will reveal themselves when successful products and architectures emerge from the competitive melee.

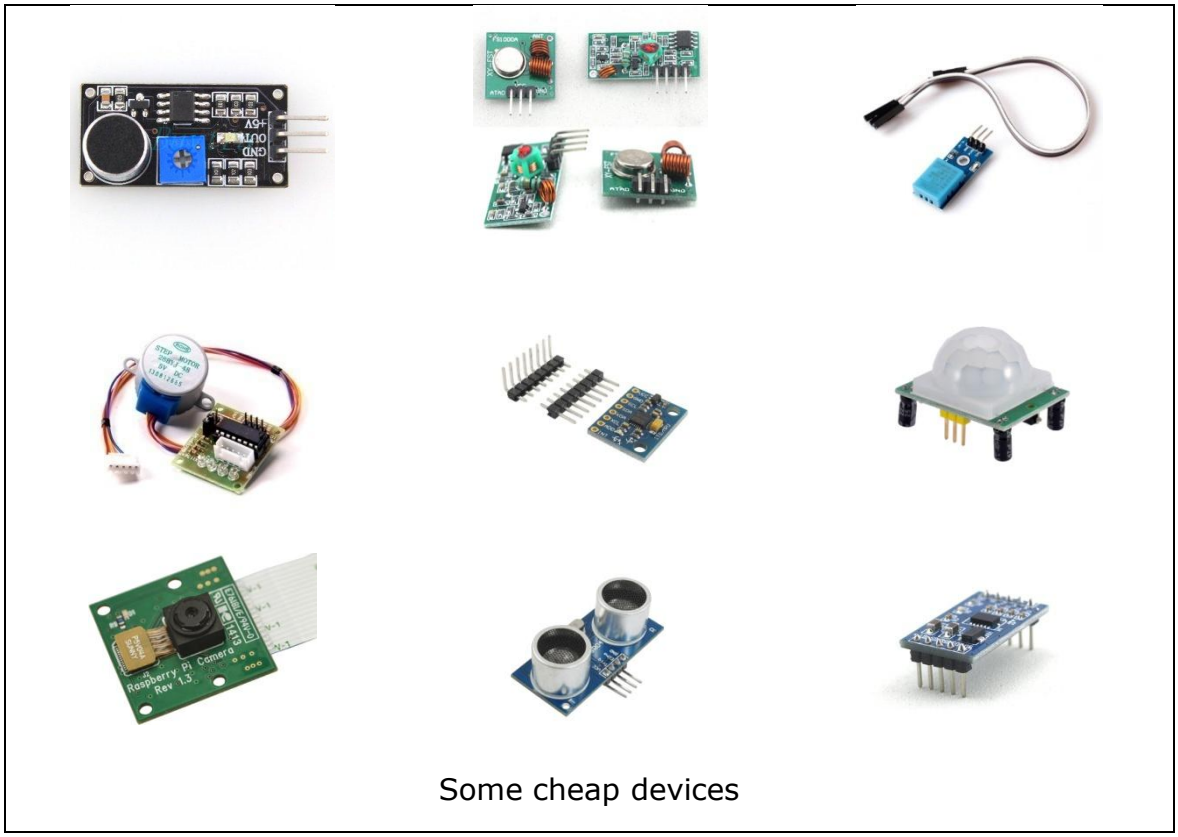
Right now, the future is being predicted with more than a little hype. The industry research companies are struggling to figure out how to make reasonable hype-free predictions. A better than average summary can be found in [1].

In this article, I'll look at the evolving architecture of the IoE and speculate on the emerging risks of it all.

Devices on the Frontier of the Internet

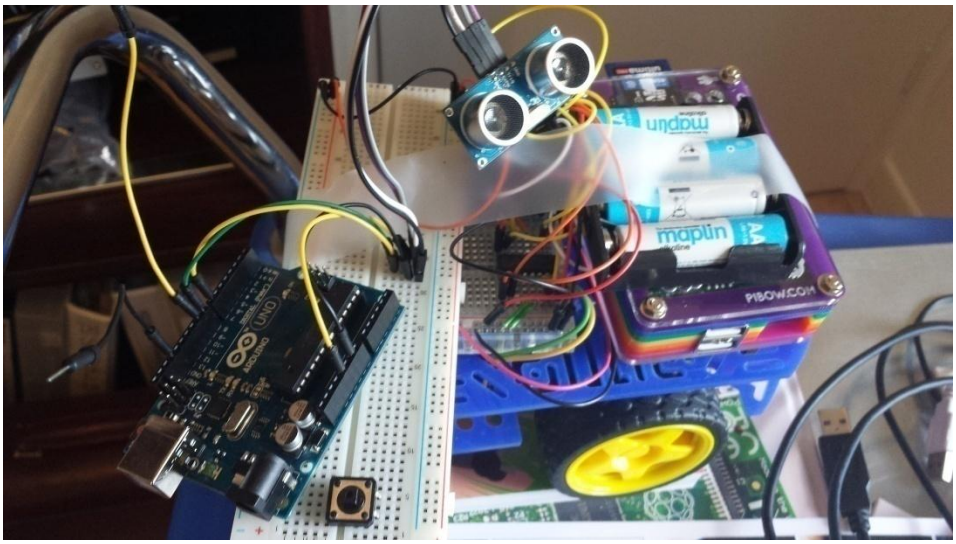
Some people predict the current internet of around 7 billion connected devices (around one device per human being on the planet) will grow to being 700 billion or even a trillion in the next 10-20 years. How can that growth in numbers be achieved and sustained?

Most of the devices that will form the IoE will be built to be as 'cheap as chips'. Most devices may actually be a single chip plus a few small components and connectors on a small (2-3cm square) circuit-board. You can buy devices that are sensors, actuators, DC motors, wireless transmitters and receivers, step motors or servos, for example, that cost between one to three dollars. Here's a selection of cheap, widely available devices. Can you guess what they are?



Some cheap devices

In the past six months or so, I've buying sensors and motors to connect to my Raspberry Pi's and Arduino boards to experiment. Through curiosity, I've been drawn into the mysterious worlds of embedded software and robots. Actually, as a techy, and being a bit of a hacker, they aren't so mysterious. But I'm getting a distinct feel for how, at the local level, these devices can be packaged into useful things.



Through miniaturisation, I expect my homebrew efforts will look clumsy when multiple devices and the software that controls them are combined onto smaller boards. I look forward to playing with those too ;O)

Overall, the large majority of devices in the IoE will be rather simple, boring sensors.

Data in Chirps

If a sensor is measuring temperature or humidity or barometric pressure, then the data it collects can be transmitted in just a few bytes every minute or every hour. Messages on the IoE are often tiny in comparison to emails, web-transactions or file transfers across the internet.

The messages from sensors might comprise a device identifier and a value. The receiving device can timestamp the data and look up the source device identifier to determine its location, device type and so on. The regular, but occasional pulses of data can be compared to a bird's chirp. The chirping analogy can be extended so that different sensor types and their message signatures correspond to the characteristic sounds of different bird species. It is for the receiving device to detect and distinguish chirps.

Control messages sent to actuators (motors, servos, switches and so on), also consists of only a small amount of data. For example, a message to turn a streetlamp on or off requires only a single bit or Boolean value.

Chirps are not Critical

Sensor devices are liable to be switched off occasionally, or fail because power is lost. Connectivity might vary and be weak or they can be damaged or destroyed by poor weather, animals, children, pets or other accidental causes. Because of this, no critical dependency can exist between sending devices and receiving devices. When a sensor fails, then the receiving device must cover the loss by using data from other sensors in the vicinity, or interpolate data from previously captured messages.

Of course, if a registered sensor does disappear off the network, an alarm could be raised at HQ to trigger a site visit by a service engineer.

Publish and Subscribe

Since devices have so little power, and functional capability, they cannot and will not maintain connectivity or communicate through a 'session'. Rather, devices are likely to register themselves and publish data to a network. Other devices on the network subscribe or listen out for messages of certain types. Messages may never be acknowledged or repeated so the one-way transmissions never become conversations. They can be compared to birdsong created to indicate a bird's territorial coverage.

Nature as a Networking Metaphor

The scale of the emerging IoE mirrors that of nature. The billions of individual ants, fish or birds in an ecosystem are not centrally controlled. Rather, they 'know their place' and they send and receive messages and act accordingly. For example, in an ecosystem where there are many species of birds, each species has its own birdsong or call pattern that it produces and tunes in to. Other call patterns are ignored. In a similar way, devices will 'chirp' their data to local networks. Most other devices will ignore these chirps. Some – at least one – will monitor and use them.

Smart-Dust

By the way, smart-dust is a term that describes a wireless network of sensors or devices that detect light, heat, vibration or chemicals. These devices are less than one millimeter across but require

antennae that are larger (to have a workable range). There are few applications as yet, but smart-dust is a stepping stone to microscopically scaled devices and of course, nanotechnology. Watch this space.

Low-Cost Connectivity

The devices at the periphery of the IoE are extremely low-cost nodes on local networks. They are often unable (and do not need) a highly-reliable, high volume and flexible software stack to perform their function. Rather, each device might contain only enough hardware and software to register its existence on a local network and transmit its brief chirps to listening devices in the vicinity. This is the argument for very low cost devices to use local, cheap, low-power, short-distance networking mechanisms. One example which may or may not come to dominate the M2M domain is OMA LWM2M [3]. Other standards exist or are emerging.

More sophisticated devices may well use IP-based networking. In these cases, some form of lightweight messaging service such as MQTT [4] (which implements a publish-and-subscribe protocol) might be used.

Although the economics of today's small devices favour non-IP based networking, it looks like sensors with a full IP stack and wireless connectivity costing a few cents may be available in a few years.

It is likely that all of these networking options will be taken up but none will dominate. One key aspect of how a local device network might be architected is whether a device needs to communicate in both directions, needs to be remotely managed and measurements are pulled rather than pushed from the device. The more the device is responsible for, the more sophisticated the networking has to be.

A range of open standards are being developed alongside proprietary products and architectures. It's not clear who or which will win the day. In Part I of this series I mentioned the better known initiatives.

Devices are 'On their Own'

Since the devices are so basic, when they power-up, they must register themselves on a network and start transmitting immediately with no human intervention. This requires that the device 'knows' what it must do much like the new-born ant, bee or termite knows that it must build the nest, find food or look after young. The servers that the devices connect to must be discoverable because as devices are implemented in the field in their millions or billions, they may not be uniquely configured – they have to 'make their own way'.

Device Traffic is Mostly One-Way

Needless to say, the most basic sensors, when powered on need to simply chirp data to the outside world, or at least to the nearest listening device. If the sensor has a single mode of operation and requires no calibration or cannot be configured or turned off, then it only needs to transmit. Simple sensor devices require only a source of power, a controller, a transmitter and a case or housing to hold the components together. They may not have the capability to listen for replies or messages from other devices at all.

Some devices may be capable of registering on a network and being remotely managed to transmit on a different channel, to assign a unique device identifier or to chirp a certain number of times per hour or every so-many seconds.

Communications are Unreliable

Simple sensors are very low power devices and might have a built-in battery of limited life or maybe a small, light-sensitive solar panel. Batteries eventually fade and the device dies. A light-powered device might be in the shade, or be covered by a leaf or temporarily lack the power to chirp. If a device goes

quiet for a while, the receiving device must still function and make decisions. The ability to lose messages and interpolate values between transmission gaps is required.

Signal Simplicity

The messages that are sent and received by devices at the periphery are usually very simple and short. There are many possibilities for the format of these very low-level messages but the data fields they contain will normally include:

- A device/chirp family identifier (the type of device)
- A device identifier (unique in the context of a local network)
- Chirp parameters (frequency, chirp serial number)
- Chirp payload (the message content)

The payload may contain any content in any format that the receiving system understands. Note that these messages may be encrypted for security, but the addition of SSL security introduces a significant overhead to what are, after all, very simple devices.

To be continued...

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

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

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

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

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

Back To Index


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

An Open Source Approach to Automated Accessibility Testing

- by Sanjay Joshi

Accessibility is a path leading to the development of application which is handy to use even for the physically challenged person with or without the aid of assistive technology.

Government, educational institutions and business today are trying to meet their obligations under the DDA (Disability Discrimination Act). And with this increased requirement to assure quality in terms of accessibility, the IT Managers are compelled to test the application they develop for accessibility. This boom or demand for the tools to check an application for accessibility has led to the development of plethora of them. Some of them involve licensing cost; others either exaggerate or belittle the problem. The tools also have the limitation for static testing, and they lack the flexibility of being molded according to the requirements of the project.

As organization grows in this world of competitive market it incurs many additional costs, so its strategic focus is always to optimize the resource cost without any negative influence on the productivity and overall turn-over. My motive of delivering this presentation is to propose a framework for an open source approach to Automated Accessibility Testing using Selenium WebDriver.

This framework addresses some of the vital limitations of available automation tools for accessibility, and is designed to be twisted according to the requirements of the project.

Why Automated Accessibility Testing?

Well!! to be precise "if we want to assure better quality of application in terms of accessibility by conducting the testing in depth, automation is a perfect add-on to manual testing for fulfillment of the purpose."

Also if the application is large, i.e. one containing many pages and navigation links, testing it for accessibility only via the approach of manual testing would become a cumbersome task. Human beings

are prone to making mistakes when they have to perform same task repeatedly, so the chances of missing a bug multiplies with the size of the application.

Moreover automation is a right approach for the organizations seeing accessibility as a secondary testing after all other features has been tested, which never gets completed or is completed just before the product release. Other benefits are:

- Reduces overall cost and provides an important assistance for regression during the internal releases of the application.
- It speeds up the process of testing as compared to manual approach.
- It can be repeated as many times as required.
- It can assist in the quick error fixing by integrating with the development environment.

What led to the designing of this framework?

I noticed a group of individuals with special abilities performing testing on certain application, and this made me curious to learn what and how they were actually testing? To relinquish my thirst for getting the answers, after a knowledgeable discussion with the group, I finally discovered that they were engaged into an interesting and beautiful world of Accessibility Testing.

I started collecting information about the different guidelines to verify, while performing a check on application for accessibility conformance. And being an engineer from automation background, my primary research was to find tools for performing it via Automation Testing. But picking a right tool for automation among the plethora of them requires significant skill and knowledge. Being a believer of "Prevention is better than cure" I pre-defined some functionality, the tool I choose should have for effective and close to complete accessibility testing. These objectives were:

- It should be capable of automating all the vital checkpoints of WCAG among the 65 proposed.
- It should be capable of testing for ARIA support.
- It should be an open source tool.
- It should be suitable for dynamic testing.
- It should be flexible.
- It should have an effective reporting system.
- It should have a scope for extension according to the future requirement.

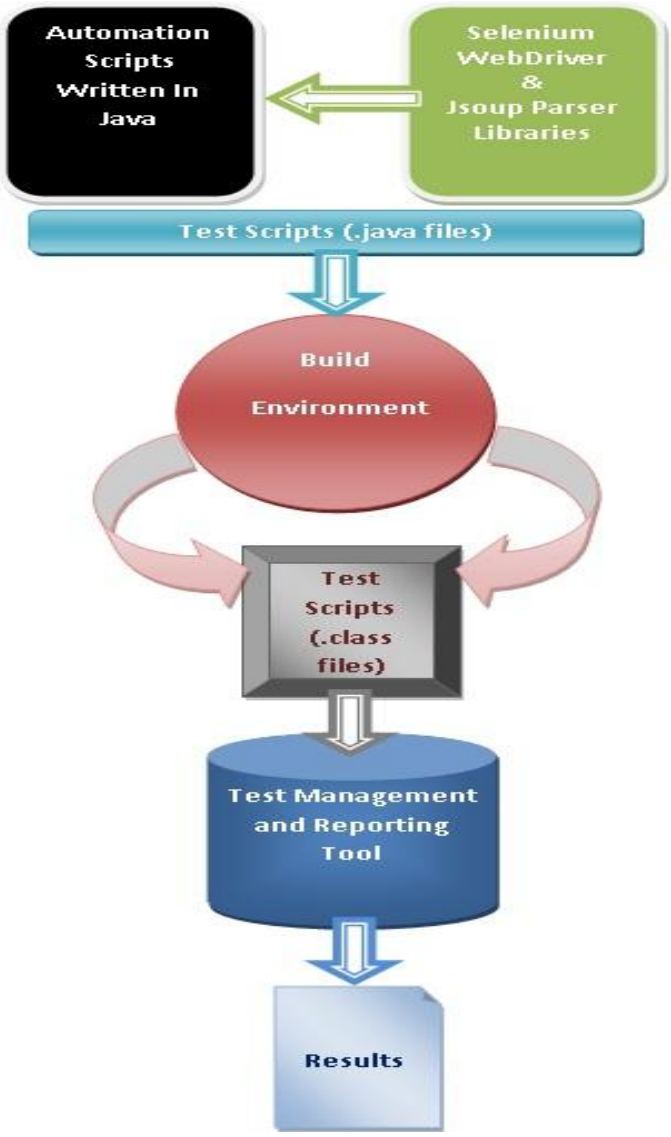
Survey Conducted On Available Automation Tools

Based on the popularity, functionality and trial period (available on paid tools), and study of six accessibility tools generated the following results:

- No tool under consideration was able to test more than 30 percent of the WCAG important checkpoints in average.
- Most of them did not test for ARIA support.
- Involved licensing cost ranging from $100 to $2000
- No tool had a support for dynamic testing.
- The tools lacked flexibility of being tweaked according to the project requirements.
- The reports generated by the tools were large in volume but difficult to be processed in any meaningful way.

Finding no appropriate tool to meet the stated specification, I leveraged the functionality of few open source tools (as they provide low cost solutions which do not put strain on client's budget); to design a framework for testing web-application and assure that it is accessible to all.

Framework Architecture



(Figure 1)

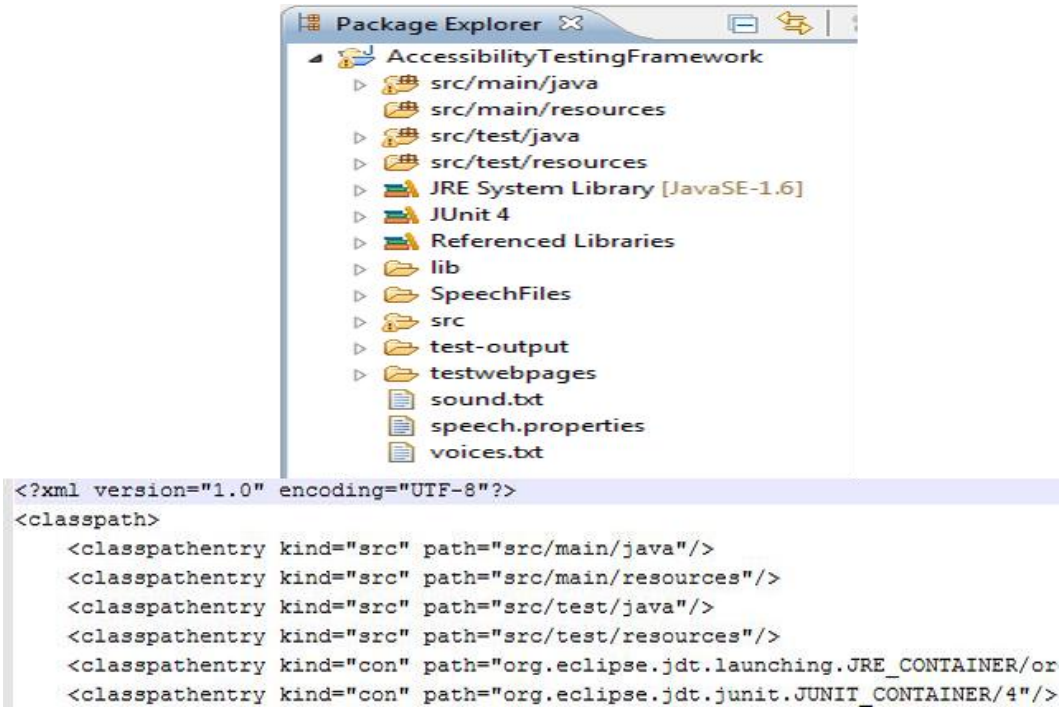
From figure 1: Test scripts are written in Java language, by leveraging the libraries of the Selenium WebDriver and Jsoup. WebDriver is used for driving the browser and performing actions on the page-objects while Jsoup is used to parse through the HTML code.

The scripts are then compiled to .class files so that they can interact with the Test Management and Reporting tool such as FitNesse. In this case Eclipse IDE can be used as a build environment. Framework also has the flexibility of using with Maven, where it can act as both Build Environment and Reporting tool. For my purpose I have used FitNesse as it provides easy-to-understand reports after test execution has completed.

The .class files then interacts with the Test Management and Reporting tool (FitNesse in this case) for executing the tests on the application. At the end, when the entire test scripts are executed, final report is generated, which can be used to improve the accessibility of the application.

Framework Structure

All the test files are placed in the folder "src/test/java", and resource file corresponding to test scripts are kept in the "src/test/resources" folder. The files containing the methods to perform the action are kept in the src/main/java folder. The framework is designed in this structure (Figure 2) to give it the flexibility of being used with both the popular reporting tools FitNesse and Maven. This structure is available directly when we create a new Maven project, but to achieve it while creating a normal java project in Eclipse, we have to modify the .classpath file of the newly created java project as shown in Figure 2.



(Figure 2)

Example of Automated Accessibility Testing Using the Framework

'Drag and Drop' feature is commonly seen in every web-application built today. But such type of usability involves end-users to drag an object from one location (using a mouse click-and-hold event) and shifts to another location, this imposes a major challenge for people with motor impairment as they have problem in making movement and remain steady. So they should have a keyboard equivalent available to allow them to use such functionality in a different way. The functionality is also difficult to use for those individuals having poor eyesight, as they face challenge in determining the appropriate start and end point. For these groups of people, it is essential to test that the functionality is available to be used in alternate ways.

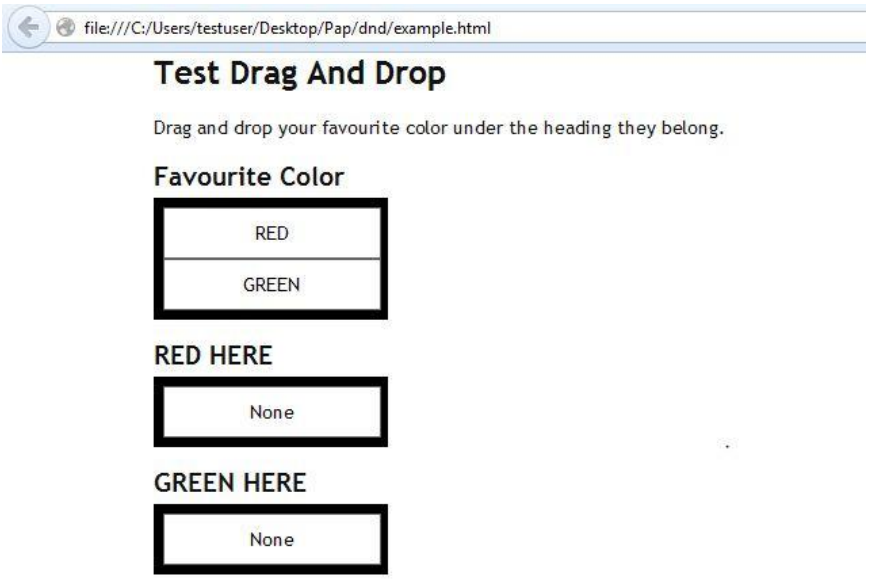
The e-learning project I am working on has a feature wherein students can drag-and-drop a file to submit it as an assignment to the instructor. The functionality is designed using WAI-ARIA guidelines for drag-and-drop affect to make it available via keyboard and address the following challenges encountered while using screen readers and magnifiers:

- Identifying objects available to drag
  - *"draggable = true" property indicates that the element is available to take part in a drag and drop operation*
- Identifying which object has been selected to drag among the available.
  - *"aria-grabbed = true" indicates the element has been selected to drag.*
- Identifying the location of the target.
  - *"aria-dropEffect" property is used to determine the target*
- Identifying the event that will occur after dropping the object on the target.
  - *"dropEffect" attribute is used to determine what happens when the object is released on the target. It can take values as;*
    - *move: The source will be removed from its current location and placed at the target location*
    - *copy: The source will be copied on the target*

To know more about these attributes visit: http://www.w3.org/TR/wai-aria-practices/#dragdrop

For the demonstration purpose here, I have used a similar web-page (Figure 3) designed using WAI-ARIA guidelines, which has a feature to drag-and-drop your favorite color under appropriate heading (RED HERE & GREEN HERE). To use the functionality using keyboard follow the following instruction:

- Tab is used to reach the color you want to move.
- The item to be move is selected using Spacebar.
- The target is selected from the context menu using up and down key.
- Press Enter to move the color to the target list.



(Figure 3)

Testing Drag and Drop Feature

In accessibility all the input and navigation is performed using the keyboard, In the framework the 'pressing of tab key' to reach an element is achieved by focusing on the active element using the WebDriver 'activeElement ()' method and thereafter using the 'sendKeys ()' function to issue Tab as 'sendKeys (Keys.TAB)' one after the another until we reach the element on which action has to be performed. Hence we are also testing here that the elements are reachable using the 'Tab Key'.

On launching the application "aria-grabbed" attribute for elements under the **Favourite Color** has value 'false', as none of it has been selected for the drag operation. This scenario is tested by fetching the value for 'aria-grabbed' attribute and comparing the actual value fetched using WebDriver with the expected value (which is 'false' in this case). Code snippet for testing this scenario;

```
public boolean _testElementIsAvailableToDrag()
{
    WebElement redColorDraggable = driver.findElement(By.className("draggable"));
    String flag = redColorDraggable.getAttribute("aria-grabbed");
    if(flag.equals("false"))
    {
        return true;
    }
    else
        return false;
}
```

When a single tab is issued, the web-element 'RED' under **Favourite Color** gets selected (Figure 4) as an element to be grabbed. Tabbing is achieved as;

```
public void _issueTabsToNavigateToAnElement()
{
    WebElement initialElement = driver.switchTo().activeElement(); //Focus is set to the first element
    WebElement focusedElement = initialElement;
    _tabThroughTheElement(focusedElement); //For issuing a single tab (use loop for multiple tabs)
}
```

(Figure 4)

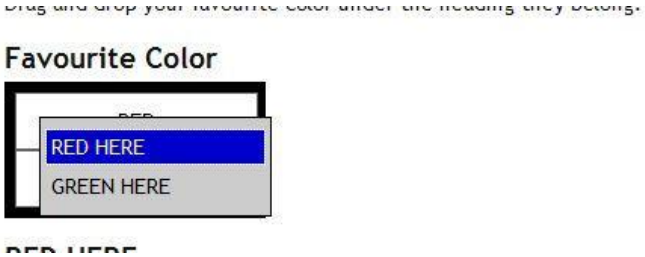
This scenario is tested by fetching the value for 'aria-grabbed' attribute and comparing the actual value fetched usingWebDriverwith the expected value (which is 'true' in this case). Code snippet for testing this scenario;

```
public boolean _testElementIsSelectedForDragOperation()
{
    WebElement redColorDraggable = driver.findElement(By.className("draggable"));
    String flag = redColorDraggable.getAttribute("aria-grabbed");
    if(flag.equals("true"))
    {
        return true;
    }
    else
        return false;
}
```

Now, we select the target by issuing a key event of the Space-bar using the WebDriver. This is achieved as;

```
public void _issueSpaceBarPressEvent()
{
    WebElement redColorDraggable = driver.findElement(By.className("draggable"));
    redColorDraggable.sendKeys(Keys.SPACE);
}
```

On performing the Space-Bar press event a context-menu (Figure 5) appears, we will test



(Figure 5)

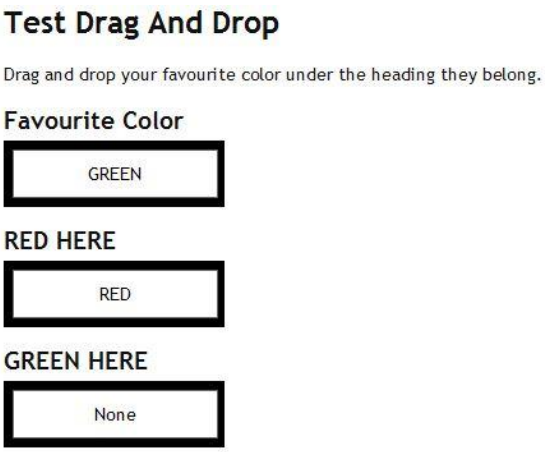
the menu for its valid 'role' attribute as;

```
public boolean _verifyMenuRoleAttribute()
{
    WebElement contextMenu = driver.findElement(By.id("popup"));
    String roleValue = contextMenu.getAttribute("role");
    if(roleValue.equals("menu"))
    {
        return true;
    }
    else
        return false;
}
```

To drop the element on the target we performing the 'Enter Key' press event as;

```
public void _issueEnterKeyPressEvent()
{
    WebElement contextMenuItem = driver.findElement(By.id("popupRedHere"));
    contextMenuItem.sendKeys(Keys.ENTER);
}
```

The object is now dropped on the target and the element under the heading "RED HERE" changes from 'None' to 'RED' (Figure 6) and the color is removed from the Favourite Color list.



(Figure 6)

To test whether the object has been dropped on the correct target or not, we check that the state of the target is changed from 'None' to 'RED'. The code snippet to achieve this is;

```
public boolean _verifyObjectIsDroppedAtTheTarget()
{
    WebElement redColorTarget = driver.findElement(By.className("redTarget"));
    String colorContent = redColorTarget.getText();
    if(colorContent.equals("RED"))
    {
        return true;
    }
    else
        return false;
}
```

As we have successfully tested a complex drag and drop feature for its accessibility, the framework can be used to put the complete application under test for accessibility.

We can also use the framework for testing dynamic scenarios such as; if the value for "alt" attribute of a cricket ball image is 'Ball'. This value does not define the type of ball (i.e. cricket or football), hence is not a valid attribute-value for this scenario.

Available automation tools will mark the test for it as passed, because they only verify if a value is available for an "alt" attribute or not. But they are not capable of verifying if it is an appropriate value.

The beauty of the framework comes here, wherein we can verify if the value for an image is an appropriate description for it, by cross-checking it against the stored values in the *.propeties* or *.csv* file present in the framework.

Key Takeaways

What you learned?

- Cost effective automated accessibility testing is possible
- Automated dynamic accessibility testing can be conducted
- You got an open source framework for accessibility testing of the application
- Solution to the key challenges encountered while using available tools

Moreover this paper is a motivation for the newbie's and experts of the automation testing domain to design such framework and tools using the open source technology for making accessibility testing of complex application an easy task. Experts are also welcome to extend the proposed framework in terms of both functionality and coverage.

Road Ahead

The framework at present is capable of automating all the key accessibility features like:

- Issuing of Tab Key to reach a particular element
- Testing the resizing of the application when Ctrl+ and Ctrl- key is pressed
- Important WAI-ARIA guidelines from each of the category;
  - Perceivable
  - Operable
  - Understandable
- Testing an image for appropriate alt attribute value.
- Testing the DOM structure.
- Testing the 35 checkpoints of WCAG 2.0

With the advancement in the screen reader technology, they are now capable of reading a document available in the form of image. But no test tool can extract text from an image and validate it for spelling correctness. Also present day application has most of the data stored in the pdf format, but no available tool tests these pdf for accessibility. Also, there is no screen reader simulation test available in any of the tool, which can assure how the application will be interpreted by the Screen Reader. In the road ahead I will be working on the framework;

- To integrate an effective OCR library in the framework, to test the image data for accessibility
- To integrate a 'pdf parser', to test the pdf in the application for accessibility
- To integrate a screen reader simulation test.

Conclusion

Automated accessibility testing can help in determining several issues which are not possible to detect by manual testing. It also increases the test coverage of the application and assists in releasing a quality product. It reduces the manual effort when thousands of pages are to be regressed before the release day.

It also has its own limitation, and no automation tool can completely test an application or completely replace manual effort. Also no single tool is sufficient to test all the functionality of an application.

Acknowledgement

I would like to thank all my friends and colleagues for laying me their helping hands at times when I got stuck.

References

http://seleniumhq.org/documentation/

http://www.w3.org/TR/wai-aria/

http://www.w3.org/WAI/WCAG20/glance/

http://www.oneguidelineaday.com/

http://dev.opera.com/articles/view/introduction-to-wai-aria/

Sanjay is an IT professional with experience, specifically in Automation Testing, Manual Testing, Quality Control and Risk Assurance. He has worked on large delivery projects based on E-Learning domain, financial domain and E-Commerce domain for different clients. His expertise includes designing UI based test automation framework for Mobile and Web applications.

Sanjay's passion is talent development and conducting training sessions on automation testing. He has also spoken at different conferences.

He is currently working as QA Engineer with Tavant Technologies India Pvt. Ltd.

Back To Index

Considerations for Automating Data Warehouse Tests

- by Wayne Yaddow

A characteristic of data warehouse development is the frequent release of increasingly high quality data for user feedback and acceptance. At the end of each iteration of data warehouse ETLs, data tables are expected to be of sufficient quality for the next ETL phase or final production deployment. This objective requires a different approach to quality assurance methods. Foremost, it means integrating QA efforts and automation into our ETL iterations.

Essential to integrated ETL testing is test automation. Manual testing is not practical in a highly iterative and adaptive development environment. There are two key problems with manual testing.

First, it takes too long and is an inhibitor to the delivery of frequent working software. Teams that rely on manual testing ultimately end up deferring testing until dedicated testing periods, which allows bugs to accumulate.

Second, it is not sufficiently repeatable for regression testing. While we seek to embrace and adapt to change, we must always be confident that features that were "Done, done!" in a previous iteration retain their high quality in light of the changing system around them. Test automation requires some initial effort and ongoing diligence, but once technical teams get the hang of it, they can't live without it.

Integrated automated testing in database development presents a unique set of challenges. Current automated testing tools designed for software development are not easily adaptable to database development, and large data volumes can make automated testing a daunting task. Data warehouse architectures further complicate these challenges since they involve multiple databases (staging, presentation, and sometimes even pre-staging); special code for data extraction, transformation, and loading (ETL); data cleansing code, and reporting engines and applications.

"Data warehouse test automation" can be described as the use of tools to control (1) the execution of tests, (2) the comparison of actual outcomes to predicted outcomes, (3) the setting up of test preconditions, and (4) other test control and test reporting functions. Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.

Although manual ETL tests may find many data defects, it is a laborious and time consuming process. In addition, it may not be effective in finding certain classes of defects. Data warehouse test automation is

the process of writing programs to do testing that would otherwise need to be done manually. Once tests have been automated, they can be run quickly and repeatedly. This is often the most cost effective method for a data warehouse that may have a long maintenance life because even minor patches or enhancements over the lifetime of the warehouse can cause features to break which were working at an earlier point in time.

Test automation tools can be expensive; they are usually employed in combination with manual testing. It can be made cost-effective in the longer term, especially when used repeatedly in regression testing.

What to automate, when to automate, or even whether one really needs automation are crucial decisions which the testing (or development) team must make. Selecting the correct features of the product for automation largely determines the success of the automation. Automating unstable features or features that are undergoing changes should be avoided.

Today, there is no known commercial tool not set of methods/processes that can be said to represent total end-to-end data warehouse testing. Informatica, Microsoft SSIS, and other major ETL tools do not promote any one tool as the answer to data warehouse automated testing.

Common Data Warehouse Test Automation Objectives

- Automate as many data warehouse testing cycle activities as possible.
- Verify data across all points during the ETL process.
- Verify all data, not just a subset or sample.
- Verify complex data transformation rules.
- Reduce manual testing workload which could be thousands of SQL scripts.
- Identify mismatched and missing data.
- Profile the performance of your data warehouse.
- Focus on numeric financial information for compliance or financial reporting.
- Reconcile and balance information between input and transformation processes.
- Validate information between the source and its final point in the data warehouse.
- Provide assurances to auditors that all financial information residing within a data warehouse can be trusted.
- Develop a regression test suite for system testing and production monitoring.

Automation of these checks may also be required due to the complexity of processing and the number of sources that require checks. Without automation, these checks would be costly or in some cases impossible to do manually.

Data Warehouse Processes—Targets for Automation

- File/data loading verification
- Data cleansing and archiving
- Load and scalability testing
- Application migration checks
- Extract, transform, load (ETL) validation and verification testing
- Staging, ODS data validations
- Source data testing and profiling

- Security testing
- Data aggregation processing
- BI report testing
- End-to-end testing
- Incremental load testing
- Fact data loads
- Dimension data loads
- Performance testing
- Regression testing

For most data warehouse development, the ETL test processes are primarily responsible for the data quality. Hence the selection of an appropriate ETL tool (e.g., Informatica, in-house developed) is a serious concern for an organization.

ETL tools and processes may contain a variety of the following components or systems. Most are reasonable targets for automated testing.

- Aggregate building system: For creating and maintaining physical database structures.
- Backup system: Responsible for backing up data and metadata.
- Cleansing system: Commonly a dictionary driven system for information parsing. For example, names and addresses of individuals and organizations, etc.
- Data change identification system: Used to record source log file reads, source date, and sequence number, etc.
- Error tracker and handling system: For identifying and reporting ETL error events.
- Fact table loading system: It is equipped with push/pull routines for updating transaction fact tables.
- Job scheduling system: For scheduling and launching all ETL jobs.
- Late arriving fact and dimension handling system: For insertion of fact and dimension records that because of some reason have been delayed in arriving at the data warehouse.
- Metadata manager: For assembling, capturing and maintaining all ETL metadata and transformation logic.
- Pipelining system: Required for implementing and streaming data flows.
- Quality Checking System: Responsible to check the quality of incoming data flows.
- Recovery and restart system: Responsible for restarting a job that has halted.
- Security system: Responsible for the security of data within an ETL.

- Source extract system: Includes source data adapters along with push/pull routines for filtering and sorting source data.

- Surrogate key pipelining system: Typically a pipelined, multithreaded process for replacing natural keys of incoming data with data warehouse surrogate keys.

Examples of Scenarios Used by Automated ETL Tools

Automated data warehouse testing tools may operate with the following sequence of tasks.

- Client routine sends request to server to execute test.
- Server routine assigns an agent to retrieve results from the source system for the test.
- Server assigns another agent to retrieve the corresponding results from the target system.
- The initial agent notifies server that the source information is retrieved.
- The other agent notifies the server that the destination result is retrieved (may complete before number above).
- Server assigns any available agent to compare the results and report any errors.
- Agent returns to the server the outcome of the test case.
- Server notifies all clients of the results of the test.

Deciding Which ETL Components to Automate

The trick is to figure out what needs to be automated, and how to approach this task. An array of questions must be considered in the course of automating tests such as:

- How much will it cost to automate tests?
- Who will do the automation?
- What tools should be used?
- How will test results be displayed?
- Who will interpret the results?
- What will script maintenance consist of?
- What will happen to the test scripts after the launch of the application?

Not all the areas of DWH testing can be automated, but some of critical testing involving the data synchronization can be achieved using automating data warehouse solution.

A primary objective of automated data warehouse testing is to cover the most critical part of data warehouse which is synchronization or reconciliation of source and target data.

Automation Challenges

- Large number of tables and records.
- Multiple source systems involved.
- Testing the data synchronization for all these tables.
- Testing Business Objects reports through automation.

Common Solutions

A start at ETL test automation can be divided into three major areas.

1. Count Verification: Initially all the tests for source to target mapping and Incremental Load for verifying initial/incremental/differential data count can be automated.
2. Data Verification: Secondly, for source to target mapping and Incremental load data verification tests can be automated.
3. Verifying Data Types: Lastly, logic for checking of the data types of source and target fields can be introduced.

Advantages

Reusability: These automated tests can be reused with minimum efforts for regression testing when newer versions of the source systems are introduced.

Automated testing tools are most helpful when used in the following circumstances:

- Repetitive, mundane tasks. As ETL processes are developed, they may go through many versions, each of which must be tested. By creating test scripts once, automated testing allows reuse for future versions.
- Regression testing. Each time a change is made to data or business rules, testers using a manual approach have to go through each function and make sure it does not affect any other part of the system. Automated regression testing uses test cases to do this quicker.
- Multiple combinations In increasingly complex environments where many scenarios must be examined, automated testing can rapidly run many test case combinations using a single script.

The Benefits of Automated ETL Testing

- Executes test cases faster. Automation can speed up test case implementation

- Creates reusable test suites. Once test scripts have been created in an automated tool, they can be saved for easier recall and reuse, and multiple testers often will have the capability and skills to run them.
- Eases report generation and records test results. An attractive feature of many automated tools are their ability to generate reports and test records. This capability can provide an accurate representation of the state of the data, clearly identify any defects, and be used in compliance audits.
- Reduces personnel and rework costs. Time that would be spent on manual testing or retesting after fixing defects can be spent on other initiatives within the IT department.

Limitations to Automated Testing

- Does not eliminate manual testing. Although automation can be used for many test cases, it cannot totally replace manual testing. There are still some complex cases where automation may not catch everything, and for user acceptance testing, end users must manually run the test. Therefore, it is important to have the right mix of automated and manual testing in the process.
- Cost of tools. Automated testing tools can be costly, depending on their size and functionality. At a first glance, the business may not see this as a necessary cost; however, the reusability alone can quickly turn it into an asset.
- Cost of training. Some automated tools can be complex to use and may require training to get started. Testers must be trained not only on the software, but also on the automated test planning process.
- Requires planning, preparation, and dedicated resources. The success for automated testing is highly dependent on clear requirements and careful test case development before testing starts. Because each organization, scenario, and application can be unique, an automated testing tool will not create test cases. Unfortunately, test case development is still a manual process.

To be continued in next issue...

Wayne Yaddow is a computer testing professional. He worked with IBM as a Z/OS mainframe operating system developer and tester for fifteen years. After IBM, he continued his career as a QA consultant, working primarily in the financial industry in NYC with firms such as Standard and Poor's, Credit Suisse, Citigroup, and JPMorgan Chase. Wayne focused on business intelligence, data warehouse, data migration and data integration testing projects. As a contributing author to Better Software, TDWI Business Intelligence Journal and a participant with The Software Testing Professional Conference (STP), he has gained a reputation as a well-informed database and data warehouse quality assurance analyst. Most recently, Wayne teamed with Doug Vucevic to write the book, "Testing the Data Warehouse", 2012, Trafford Press.

Road to Test Automation

- Part 1

- by Georgios Kogketsof

One of the most interesting areas in the software quality assurance profession is the automated software testing. An automated test can be implemented by using a record and replay tool such as Selenium IDE or by coding the automation script using Selenium with any of the programming languages such as java, Groovy, C++, .Net and Python

In this series of articles we will focus on how to create automated test scripts using a programming language to test Web applications. The reference tool would be the Selenium RC and/or Selenium Webdriver (I consider them to be two different testing tools just because they are based on totally different philosophy to perform the tests).

I will try to present a way to create automation scripts and I will present methodologies, techniques and solutions to achieve:

- Code Maintainability
- Test Maintainability
- Code Efficiency
- A clear path to follow in order to create your own testing framework

At the end of this series I would be glad if I had helped people to avoid the same mistakes that I did, and extremely happy if they have found answers to their automation problems.

When we started our automation-testing journey with a thought that our programming skills would be enough to get us a walk in the park, little we knew then. The first indication for a bumpy ride came when a little change in the application created an avalanche of changes in our testing scripts, not that our scripts were fragile; just that this small change had to be implemented to 100 scripts. The immediate thought was; what would happen if we had 600 scripts? So we needed 'a plan' to handle such situations.

To overcome the maintainability issues we decided to use widely -accepted guidelines. These guidelines are based on concrete methodologies such as page objects (https://code.google.com/p/selenium/wiki/PageObjects) and adopted by the majority of test teams around the world.

The primary focus is to guide a test team in producing maintainable test code, which does not break when AUT (Application under Test) code changes. In an In agile project QA teams often overlook maintainability because of the "agile" pace; a successful automation engineer should find equilibrium and ensure that the product quality desired by the customer - or dictated by the project manager - is delivered without sacrificing best practices that ensure automation robustness.

The first guideline to ensure maintainability is the introduction of abstraction layer to our test design. The abstraction layer will allow us to separate the test functionality from the actual implementation of the automation code. One of the advantages of the abstraction layer is the freedom of trying different test tools (Selenium RC, WebDriver) without affecting the actual test. The proposed layering model is shown in figure 1.

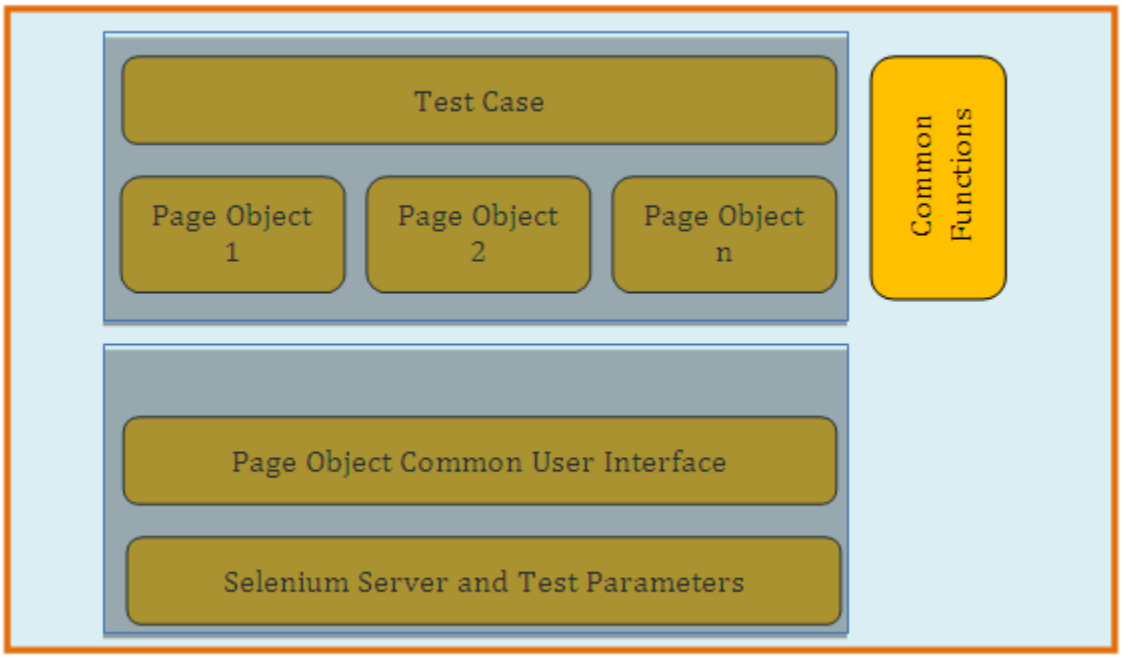


Figure 1 Abstraction Layering Model

The above model consists of two sections. The sections are separated to show that the top section is test case dependent while the bottom one not.

The top section consists of the actual test cases where the test functionality is defined and the page objects where the element locators are stored as well as the user interface actions. Common functions supporting the test cases are vertically layered but mostly concentrated at the bottom part of the diagram.

This building block should contain your higher-level API that is comprised of building blocks such as:

- Patterns (Builder pattern, Factory pattern, State pattern, etc.)

- Methods that have many interactions with the system, in such a way that are reusable (e.g. data-driven testing).

The bottom section consists of two layers. The first layer is used for defining page object common user interface functions that will assist in building page objects more quickly. This layer would contain methods for user actions such as click, select, drag, get/set input text etc.

The second layer is used to separate the selenium session and the test parameters from the rest of the test.

The aforementioned automation model holds some interesting points:

The first point is that the bottom section is independent of AUT and thus can be decoupled from the model. Decoupling them suggests the creation of a framework and that is what we did. In future articles I will explain how we took the first two layers and implemented a testing framework.

The second point is that the proposed model can be used with different testing tools such as Selenium RC or Webdriver or even testing frameworks like Capybara with minor adjustments.

Lastly a point worth noting is that the usage of page objects is inadequate: as I will explain in a following article, it does not guarantee maintainability if it is not used in a right way.

Talking about page objects – what a page object is can be easily found by a quick search on Google and will not waste your time reproducing here – I will quickly elaborate on how to efficiently create them. Multiple occurrences of the element locators within the page objects create a big maintainability problem. For example an input element having 3 actions (type text in a text box, get text from a text box, is text box present) has its locator used 3 times in the page object; one for each action as shown in figure 2.

```
public void inputTextBoxValue(){
        selenium action on element locator = some xpath or css ❶
}

public void getTextBoxValue(){
        selenium action on element locator = some xpath or css ❷
}

public void isTextBoxPresent(){
        selenium action on element locator = some xpath or css ❸
}
```

Figure 2 Page Object containing textBox

An update of the element's locator would result in updating the page object in 3 places; amplify that with the number of element of the page object and we still have a great maintenance effort.

With the proposed layered model the above problem is solved by declaring the methods representing the user actions to the page object user interface layer and leaving the element locators to the current layer as shown in figure 3.

```
String x = element locator                                    ❶

public void inputTextBoxValue(){
        action on x
}

public void getTextBoxValue(){
        action on x
}

public void isTextBoxPresent(){
        action on x
```

Figure 3 Page Object containing textBox

As it is evident, with the proposed implementation the maintainability of effort is reduced by updating the element locators in a specific place (page object), only once within the page object. Although this is one of the primary goals in implementing the test scripts with the proposed testing framework, to go one step further we have to not only define our locators to only at one place and only once but to make them robust (e.g. using css or id attribute). For a concrete example check the next articles in the series…

To be continued in next issue...

Georgios Kogketsof is a full time test engineer. In his 15 years of testing experience has worked in multinational, multicultural testing projects for major corporations in the defense industry and in digital marketing. George's expertise as a developer and as a tester fused in creating a hybrid model of a test engineer proved to be extremely effective in automation testing. Over the years George and his testing team have developed a methodology for creating fast, maintainable scripts for automation testing web apps. George is proud in his involvement in the creation of the open source testing framework Stevia.

Visit his blog at: http://seleniumtestingworld.blogspot.gr/

Get Stevia at: https://github.com/persado/stevia

CONTINUED FROM PAGE 12...

I think the most common mistake is to let outsiders who don't really understand testing dictate the way we test. Most often it is project management, who fixate on the process and want a series of neat milestones they can slot into their project plan. They want metrics they can track, and they want the testing packaged into neat chunks (like test cases). Production of a series of "work products" becomes the most important aspect of the test manager's job. It's easy to lose sight of the real job.

What is your opinion about testing standards and their implementation in organisations?

There are so many problems it's hard to know where to start. At best generic standards like ISO 29119 are an irrelevant distraction that should be ignored. However, I don't think that's a realistic option. Their status as standards implies a level of consensus and credibility that will encourage governments and big companies to insist that they are followed, that suppliers comply. In reality, the standards focus on one approach, a document driven approach that encourages testers to concentrate on the documentation rather than the product. It's a familiar problem I've seen repeatedly.

You can find slides for my CAST talk here . And the Blog article here

You have rich experience of auditing projects, processes. What good/bad happen when organisations focus more on creating auditable testing artifacts?

You get what you focus on. If the focus is on the documentation then you get good quality documentation, even at the expense of the quality of the testing. As an auditor I always disapproved of auditees who seemed to be doing work to cover themselves in a future audit rather than to do the best job they could in the present. Sometimes people would do entirely unnecessary and wasteful things because they wrongly expected the auditors to want them.

In fact, there's an argument that artefacts should never be produced solely for auditors.

If there's a genuine need then it's needed by users, or for the smooth running of the project, or for regulators, or as evidence in a possible legal case. If it's needed only by the auditors then something is probably wrong. Both the auditors and the auditees should be sitting down to talk about that.

What is your opinion about number based test metrics? Instead, what parameters would help organisations to manage testing better if they start measuring them?

I recently saw a tweet from Nassim Taleb; "all metrics are wrong, most are harmful, some are useful. There is a huge amount of damaging confusion about numbers and metrics. Managers think they're counting the things that matter. In reality they are almost always counting something that is only a proxy for the things that really matter, but can't be counted. In particular, I really don't like the focus on counting test cases. That's not helpful and I've seen testers being forced to structure their testing in a way that allows easy counting of test cases, even though it produces confusion in the text execution where simple, discreet test cases might not be a useful concept.

I'd recommend Michael Bolton on this subject. Just search for metrics and Michael Bolton. He explains the problem clearly.

As for possible useful parameters, a good starting point would be for project managers and stakeholders to maintain a clear distinction between measuring the testing and measuring the product. Measuring the testing itself is rarely helpful. Measuring the product can be, depending on the context. I spent a lot of time working with insurance financial systems.

Then it did make sense to use numbers to measure the product, because the product was money and the systems had to track and process money. We had to be sure that they were operating to acceptable tolerances. These tolerances had to be analysed, discussed, agreed and tracked. That was the context in which we were working. It wouldn't have been helpful in a different context.

A good test manager is someone who…..

…is brave enough to do the testing, and tell it like it is. A good test manager should never be happy with merely following the process and ticking off the milestones on the project plan.

Things that you would like to see changed about testing in near future would be?

Integrity, Insight, Confidence, Experience and Communication skills. All of these are vital.

What are those skills you would ask fresh testers to get good at?

It partly depends on their background. If they have a background in development, or significant experience in coding, then I'd strongly recommend that they keep their coding skills up to date. I also think it's valuable to become knowledgeable about the business areas you're testing in. Obviously you also need to learn testing techniques and skills, and how to think as a tester.

However, the really important and undervalued skills are the soft skills of dealing with people, communicating and writing your ideas and findings clearly. It's a priceless skill, and often organisations don't realise how important it is till they find that they don't have people with those skills in the roles where they are needed. Testers have to be able to tell a story. If they can't do that then much of their good testing work can go to waste.

Your message to our readers would be?

The fact that they are reading Tea-time with Testers shows that they are interested in the profession of testing and want to get better.

Keep doing it! Ultimately your career depends on what you are prepared to learn, and how capable you are at adapting to change and picking up new experience and skills. We all need to keep at it, and magazine like Tea-time with Testers are vital.

You have written for Tea-time with Testers on multiple. How do you feel about this association with us? We would like to know your feedback for our further improvement.

It's been an honour to write for the magazine. I'm grateful for the chance to spread my ideas.

I think it's very important to keep providing a platform for testers from the Context Driven School, and indeed all testers who truly think about their profession.

Thank you, James. It was great talking to you.

Happiness is….

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

Like our FACEBOOK page for more of such happiness

https://www.facebook.com/TtimewidTesters

STM AGAZINE

KEEP
CALM
IT'S
COMING
SOON

Be the first one to know. Follow us on Twitter

@SoftoolsMag

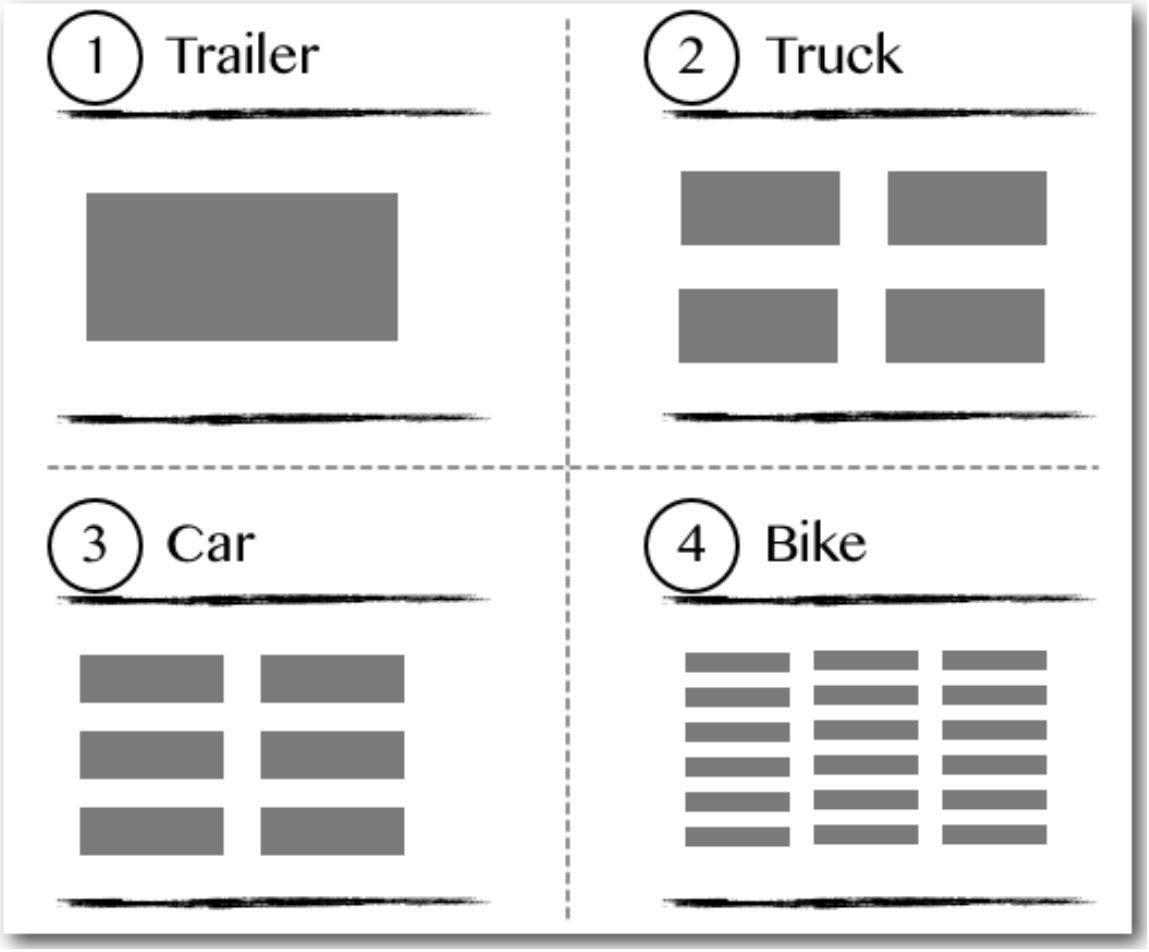
Load testing is not bombarding the system with concurrent users

"We want you to load test our application. And we want you to test with a load of 50k concurrent users. Can you do this?" asked the gentlemen from a company developing an interesting portal application said during the sales call.

And my first reaction was 'Whoah, hold your horses, load is not just about mere number of concurrent users'. And then I started my explanation of what the objective of load testing is and how they should look it. After half an hour, they understood. They understood that this was not a simple web site where you bombard it bunch of hits from users, but a web application that does various operations. And evaluating the load handling capability of the application requires subjecting it to a mix of various operations with some of them being concurrent

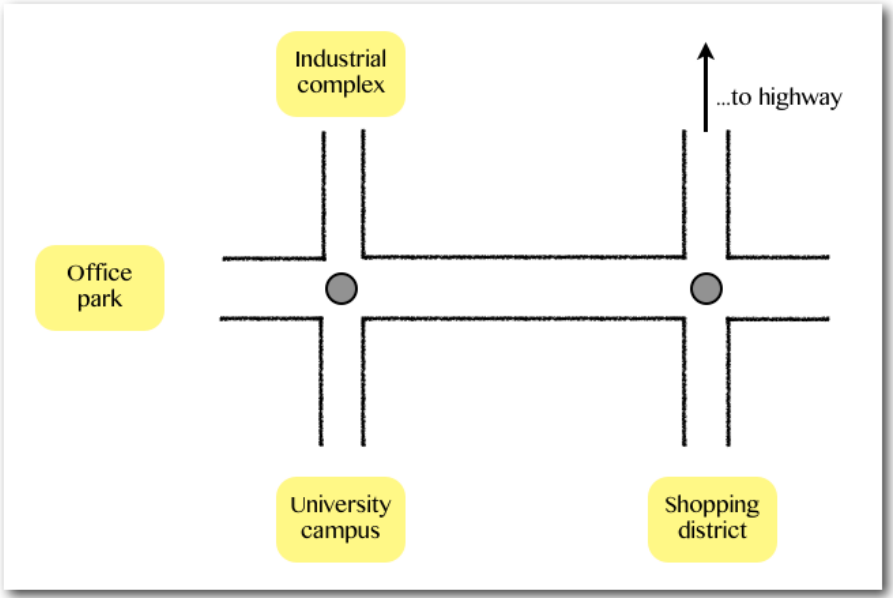
Allow me to share this with you.
Let us start with a simple definition of 'Load'. Load is the quantum of work to be performed. And load testing is about assessing if the given quantum of work can be done well in the stipulated calendar time with the given resources. Now what is work? It is various operations that are done by the system. And different end users perform different types of operations.

Consider the analogy of traffic system, of assessing the load handling capacity of a road in a city. Is it merely the number of vehicles it is able to handle at any time? Well it depends on the types of vehicles. The vehicles may be trailers, trucks, cars or bikes. The carrying capacity of the street would be much higher if the vehicle is a bike instead of trailer.



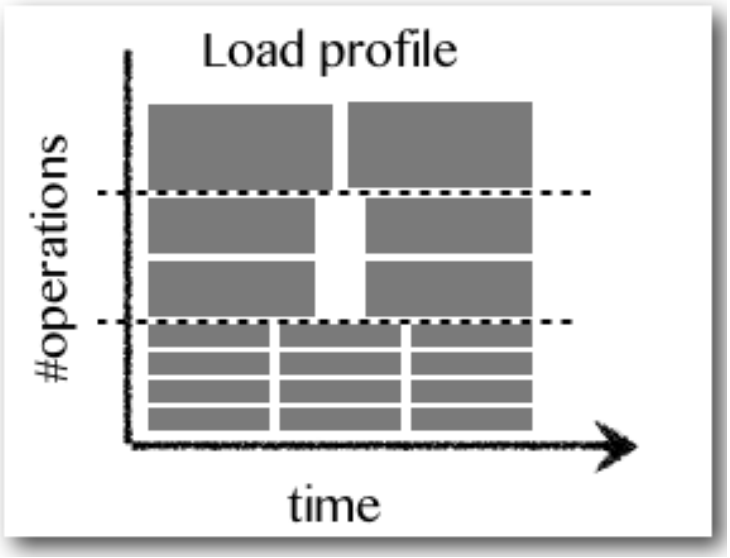
In reality, traffic is a mix of the various types of vehicles - some trucks, cars, bikes and maybe trailers. And there could be one or more vehicles of a given type (car/truck …) that may pass through the road at a point in time. So evaluating the load handling capacity of the road requires us to understand the traffic situation at different parts of the day/week.

Therefore the kinds of vehicles that pass through this road depends on the what areas this road connects and therefore the types of users it services. Trucks and trailers from the highway may use this road to go to the industrial complex during the late evening hours/night, whereas students on bikes may use the road to go/from university campus to residential complex /shopping district during the day, while working professionals may clog the road with cars during the morning/evening hours.

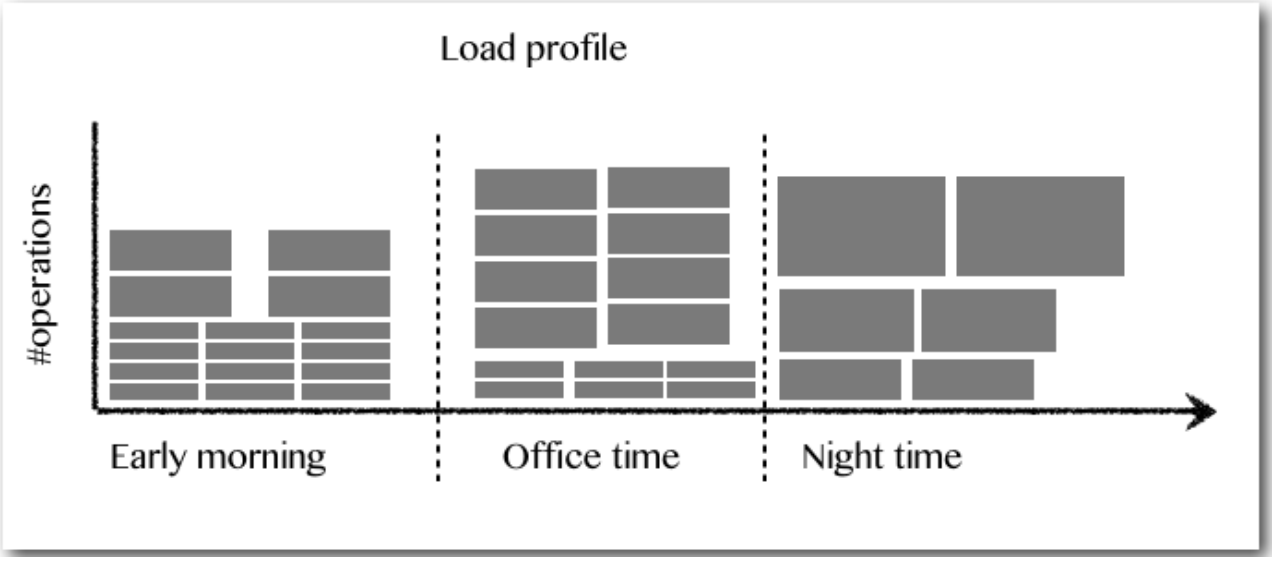


So coming back to our web application, load testing is about subjecting it to a mix of the various operations from a various types of users with some of them being concurrent and then assessing if they can indeed be processed with the given resources. That is, can the road allow the various users to go their destination safely and on time? So just bombarding the application with a bunch of users is not meaningful and can give you false alarm that the system is incapable of handling load.

Remember that a web site has only two major operations - GET and PUT and subjecting a web site to concurrent hits (a simple mix of GET and PUT) is most often good enough. And this is not true with a web application.



Now understand that a single user application can be also subjected to load and concurrency is not a pre-requisite for load testing. After all load testing is about assessing if a system can process the requisite number of operations in a given time with the given resources. And each operation may have different demand of resources and may take different times to complete. And simulating this mix and creating such a operational (or workload) profile representing different times of day/week/month/year is key to meaningful real life evaluation of load handling. Designing load test scenarios HBT (Hypothesis Based Testing) is based on this technique "Operation profiling".



So when somebody tells you perform load testing by merely bombarding the system with users, remember "The road analogy" and "Operational profiling".

When you are stuck in a traffic jam, it may be good fun to estimate the load then by applying operational profiling! On a lighter note, I do not care about the road nor traffic jam as I cycle. I squeeze through the stationary vehicles and smile at the fretting humans inside the powered vehicles. Wicked eh!

Cheers.

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

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

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

Advertise with us

Connect with the audience that MATTER!

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

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

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

Well, hear directly from them.

And the **Good News** is…

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

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


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

in next issue

our family

Founder & Editor:

Lalitkumar Bhamare (Pune, India)

Pratikkumar Patel (Mumbai, India)

Lalitkumar

Pratikkumar

Contribution and Guidance:

Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)

Jerry

T Ashok

Joel

Editorial| Magazine Design |Logo Design |Web Design:

Lalitkumar Bhamare

Cover page image - A Thoughtful Eye

Core Team:

Dr.Meeta Prakash (Bangalore, India)

Unmesh Gundecha (Pune,India)

Dr. Meeta Prakash

Unmesh Gundecha

Online Collaboration:

Shweta Daiv (Pune, India)

Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)

Kiran Kumar

Chris

Romil

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

To get **FREE** copy ,

Subscribe to our group at

Google™

Join our community on

facebook

Follow us on

www.teatimewithtesters.com