

THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

JAN-FEB 2015 | YEAR 4 ISSUE XII

Jerry Weinberg

The Three Great Obstacles to Innovation

Lorinda Brandon

Ready! API by SmartBear Software

Rajesh Mathur

LinkIn PDCA for Testers, Are you Ready?

Rama Komarabathini

Pair Testing: Doing it in Agile projects

Jim Holmes

Master the Essentials of UI Test Automation

T Ashok

"Friction-less Testing" for Developers

Over a Cup of Tea with Ben Kelly

Joel Montvelisky

Letter to a Starting Tester

Georgios Kogketsof

Road to Test Automation



Master the UI Test Automation Essentials

Webinar: March 25, 2015, 10 a.m. ET



Real-Life Test Automation

A few real-life guidelines in planning and execution can help your organization ensure UI automation projects will have the best chance at succeeding and help your teams deliver high-quality software.

Join UI automation experts Jim Holmes and Dave Haeffner for this information-packed webinar that will give you actionable steps to take in all your UI test automation projects. You'll learn tips on planning, building your team, creating maintainable test suites and, most importantly, how to ensure your stakeholders and leadership have the right information to make critical business decisions.

About The Presenters:



Dave Haeffner
WRITER,
ELEMENTAL SELENIUM

Dave Haeffner writes Elemental Selenium, a free, weekly Selenium tip newsletter read by thousands of testing professionals. He's also creator and maintainer of ChemistryKit. He has helped numerous companies implement automated acceptance testing successfully.

Dave is also a founder/co-organizer of the Selenium Hangout, and has spoken at numerous conferences and meet-ups about automated acceptance testing.



Jim Holmes
OWNER/PRINCIPAL,
GUIDEPOST SYSTEMS

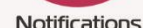
Jim is the owner/principal of Guidepost Systems. He has spent time in LAN/WAN and server management roles, as well as many years helping teams and customers deliver great systems.

Jim has worked with organizations ranging from startups to Fortune 100 companies, to help them deliver better value to their customers. Jim has been in many different environments but greatly prefers those adopting practices from Lean and Agile communities.

Register



GET A FULL VIEW INTO ALL YOUR DATA



TAKE THE FIRST STEP IN SAYING:

YES TO:

- Real time reporting
- Instantaneous data gathering
- Objective, vetted data
- Comprehensive data
- Data that mirrors your process
- Data that is always current

NO TO:

- Manual reporting
- Data warehouse projects
- Subjective data
- Missing data
- Not aligned data
- Wrong time frame data



testomaton

Quality Assurance via Automation

Software testing startup specializing in test automation services, consulting & training for QA teams on how to build and evolve automation testing suites.

SERVICES

Test System Analysis and work estimation

Review of client's system (either in development or on early stage) to produce a Test Plan document with needed test cases and scenarios for automation testing.

Tests creation and Automation

Development of test cases (in a test plan) and Test Scripts to execute the test cases.

Regression and Test evolution

Development of Regression test suites and New Features suites, including test plans and test scripts or maintenance of existing.

Architecture Consulting

Review of software architecture, components and integration with other systems (if applicable).

Trainings

Depending on the particular need, we can provide training services for: Quality Assurance theory and best practices, Java, Spring, Continuous Integration, Groovy, Selenium and frameworks for QA. For further information please check our web site.



We enjoy putting our experience to your service. Our know-how allows us to provide consultation services for projects at any stage, from small up to super-large. Due to our range of expertise we can assist in software architecture and COTS selection; review of software designs; creation of testing suites and test plans with focus on automation; help building quality assurance teams via our extensive training curriculum.

look us up: www.testomaton.com - twitter: @testomaton



TEA-TIME WITH TESTERS

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

Created and Published by:

Tea-time with Testers.

B2-101, Atlanta, Wakad Road

Pune-411057

Maharashtra, India.

Editorial and Advertising Enquiries:

- editor@teatimewithtesters.com
- sales@teatimewithtesters.com

Lalit: (+91) 8275562299

Pratik: (+49)15215673149

This ezine is edited, designed and published by **Tea-time with Testers**. No part of this magazine may be reproduced, transmitted, distributed or copied without prior written permission of original authors of respective articles.

Opinions expressed or claims made by advertisers in this ezine do not necessarily reflect those of the editors of **Tea-time with Testers**.

Editorial

The Story of Production (and layoff) – part 2

I'm glad that most of you liked my **editorial of previous issue** and some of you have also shared the opinion around solution part.

I was asked if this can be changed. Well, the answer is definitely yes!

As my friend **Albert Gareev** correctly mentioned, an Ant could have been vocal about value she added at Tiger's factory, she could have been vocal about her skills and dedication, and importantly she could have connected herself with fellow ants on jungle-net.

Now, the question is how many testers really do it? The better question would be, how many testers know how to showcase their work in a way that will earn them respect, in a way that will explain real value of their work and will make them significant for organization? Saying, "I wrote 1000 test cases", "I found 200 defects" is NOT the real measure of value we add or the work we do as testers. I remember **Keith Klain** saying this in one of our discussions that, testers are mainly responsible for the success and failure in their career, because they try to showcase their work with number of test cases they write or with number of defects they find. And I completely agree with him.

Folks, if you are really and truly concerned about your career in testing field then you must come out of this number obsession yourself first. Automation and scripting are not the only advancements and innovations in testing. Innovation also lies in the way you think for test ideas, in the way you document your test cases and in the way you report your work. Please don't ignore what appear to be 'basic' things. Innovations and advancements are needed in basic things too. And only your efforts can make it possible. You must learn the art of doing more with less.

As far as your learning is concerned, you already have free forums like **Tea-time with Testers**, **TV for Testers** and many other where you can get practical advice from experts in the field. You can get connected with like-minded testers. What really matters is your desire to make a difference.

With same desire, we had started Tea-time with Testers and we are four years old today. Thanks to all of you who have been reading, sharing and contributing to it. It's you who have helped us make it large. We will continue to contribute our little share towards betterment of testing community. All we need from you in return is helping this contribution to reach to those testers who are still looking for help and are ignorant of their own state. You got helped. It's time to pay it forward.

Will you?

Sincerely Yours



- **Lalitkumar Bhamare**

editor@teatimewithtesters.com

@Lalitbhamare / @TtimewidTesters



TEST ISTANBUL 2015

www.testistanbul.org

PERFORMANCE TESTING: HIGH PERFORMANCE SOFTWARE DRIVEN BY BUSINESS

27 March
2015

Software Testing Conference

Proudly the largest software testing conference in South East Europe and Middle East,
hosting nearly a thousand local and foreign professionals

Keynote Speakers



Goranka Bjedov
facebook



Alexander Podelko
ORACLE



Martin Spier
NETFLIX



Ian Molyneux
int Technica

Conference Fee

330 EUR VAT excluded

Contact

info@testistanbul.org
www.testistanbul.org
+90 212 290 76 62



Main Sponsor:



Event Partner:



Exhibitors:



Supportive Organizations:



QuickLook



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

Ready! API by SmartBear Software- 21

LinkeIn PDCA, Are you Ready? - 24

Letter to a starting tester - 53

In the School of Testing

Pair Testing: Doing it in Agile projects- 28

Master the Essentials of UI Test Automation -34

Road to Test Automation - 37

T' Talks

"Friction-less testing" for developers- 50

Over a Cup of Tea with Ben Kelly

Family de Tea-time with Testers



Agile Testing Software Company QASymphony Raises \$2.5MM Series A

Atlanta company proves it's ready to tackle the big problems of 24 million developers worldwide

Atlanta, GA – **QASymphony** the leading software testing solution for mid-size and large enterprises announced today that it has closed \$2.5MM in Series A funding. Led by Buckhead Investment Partners (BIP Capital), Poplar Ventures, and KMS Technology, this strategic investment will allow the company to further fund sales and marketing initiatives to accelerate its already rapid growth, including a close to 50% growth in Atlanta-based employees.

With this investment, Jamie Hamilton, a managing partner at BIP Capital, is joining the board of directors. Hamilton will bring exceptional expertise and experience working with B2B technology companies to help shape the future of this Atlanta-based company. "The team at QASymphony has proven their ability to meet the demands of a dynamic market by leading with innovation. We couldn't be more pleased to add them to the BIP Capital portfolio," stated Hamilton.

QASymphony's deep understanding of agile methodology is a key part of their success. "While our product can be used in any type of software development process, it is most efficient with agile software development, the most accepted process for building software," explained co-founder Josh Lieberman. The company's cloud based test case management tool, qTest, tightly integrates with JIRA, the most popular ALM for agile teams.

"Of the 24 million software developers and testers globally, over half have implemented an agile methodology – and that number is rising," QASymphony CEO David Keil said. "Despite this wide adoption by development teams, there has been a lack of innovative solutions in the area of agile software testing. That is a huge problem for many companies that we are uniquely able to solve."

About QASymphony:

QASymphony's quality assurance and testing tools speed up software development by reducing the time it takes to test software by 40%. With QASymphony's solutions, businesses have the visibility and control needed to ensure application quality in fast-paced Agile environments. Companies like Adobe, Barclays, BetterCloud, Salesforce Marketing Cloud, AirWatch and Vonage trust QASymphony to improve their teams' communication and productivity, while empowering them to make better software. QASymphony is headquartered in Atlanta, GA.

Are you interested in publishing the news about your own firm, tools, community and conferences in **Tea-time with Testers?**

Then write to us at:

sales@teatimewithtesters.com

with "News Enquiry"* in your subject line.

*Conditions Apply





A million dollar smile ?

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

*Adverts starting from \$100 USD | *Conditions Apply

Ben Kelly is pretty familiar name to the testers who are active in community. I got introduced to Ben via Context Driven Testing circle and he made me his fan with his blog on software testing. You guessed it right, Ben is the author of 'Testing Dead' series in Tea-time with Testers which has received a lot of appreciations. And it did make many zombie testers to retrospect.

Ben is currently working with eBay Inc. as Software Development Team Lead. I met Ben at CAST conference last year and we got to discuss some interesting stuff around testing. Read on to find it out in this exclusive interview

- Lalitkumar Bhamare

Over a Cup of Tea with Ben Kelly



Thanks for talking to us today, Ben. Your name is quite famous in the testing community. Would you like to share your testing journey with our readers?

Sure. I grew up wanting to be a programmer. In high school I had a long career planned out. The problem was I wasn't a particularly good coder and I didn't spend nearly enough time practicing to become as good as I needed to be. There were a few things that I was good at though. Video gaming was one, and spotting problems was another. The games bit I think is relevant because amongst other things playing games teaches you pattern recognition. Put pattern recognition together with the ability to spot problems and you have the makings of a decent software tester.

I had no idea that testing was a career possibility, let alone one as rewarding as mine has been. My first job after graduating from university was as a software tester. I had initially planned to bridge into programming, but abandoned that idea almost immediately. I had no real idea what I was doing, but I'd finally found a role I seemed to fit and I enjoyed it. I was fortunate enough to have a mentor there who encouraged me to explore for myself how to test things and to make improvements where I thought they could be made.

Most of the literature I found about testing involved templates and test cases and procedures, which I tried, but ultimately rejected as not really suiting what I was trying to do. It wasn't until I discovered 'Lessons Learned in Software Testing' that things started to click for me. There were other people out there that thought like I did – more than that, there was a community of thinking, skilled testers out there that I discovered I could tap into. Doing that has been one of the most rewarding experiences of my life. I've made friends all over the world and gotten to know some of the most talented software testers on the planet. These days I'm trying to share what I've learned (and continue to learn). I've been fortunate enough to travel to many parts of the globe to share what I know about testing. I'll keep doing that for as long as people find value in what I have to say.

I am big fan your blogs, Ben. Please tell our readers when did you start blogging? What was your motivation?

Like many others, I imagine, I was prompted to start writing by James Bach. I had thought up to that point that what I had to say was either completely obvious or simply not worth sharing. I was embarrassed that what I had to say wasn't worth someone else's time.

It turns out I didn't need to be. You don't have to be a recognised expert to blog about software testing. Some of the most valuable blog posts I've read are by people who struggled with a problem, or were still struggling with a problem and shared their thoughts about it. I blog sporadically. When an idea takes me then I'll tap out several posts in quick succession. I don't force blog posts though. I have many other writing projects that demand a lot of my time.

You had written an interesting article series around 'Zombie Testers'. We are curious to know more about this concept.

Sure. It started out as a kind of joke response to Alberto Savoia's 'testing is dead' meme. I thought if testing is dead, but refuses to die, doesn't that make it undead? I'm not the only person to have this thought, so I can't claim to have coined it. I guess I just took it and ran with it. The more I thought about it, the more I realised that there are certain behaviours that we fall into sometimes that are zombie-like, so I started defining zombie tester archetypes.

They describe behaviours, not people. I didn't want to demonise individuals or groups, but I did want to highlight what I saw as a lack of thinking, or rote acceptance of certain kinds of behaviours that to me are decidedly not helpful.

If I recollect it correctly, you had a big role to play when 'Testing is Dead' was discussed in community. Do you still think that it's the case with testing today as well?

I think the whole 'Testing is Dead' meme was a bit of a gimmick and ultimately not terribly useful. I think what Alberto was going for was a spin on 'Testing by rote is going to die out, skilled testing will continue and some people may not call it testing'.

... CONTINUED ON [PAGE 43](#)

Tea & Testing



with

Jerry Weinberg

The Three Great Obstacles to Innovation

SINGLE-SOLUTION BELIEF: THE NUMBER THREE OBSTACLE

We could characterize NPS as the unshakable belief in your own master intelligence. This definition makes it clear why NPS is the number two obstacle to becoming an innovative leader. The Chinese say that the first step to knowledge is a confession of ignorance. If you already know everything, how will you ever learn anything?

Even though you shouldn't believe you know everything, it does help to know something. Nobody denies that good problem solvers have to be intelligent, but lack of intelligence isn't one of the three big obstacles. We all know many high-IQ people who aren't good at solving real-life problems. Perhaps we ought to be more skeptical about the procedure by which psychologists identify "intelligence."

Recently, a science magazine ran a "Mind Benders" column prepared by Mensa, an organization of people who score in the top two percent on standardized IQ tests. Among the Mind Benders were the following two questions, which bent my mind in a direction perhaps not intended by Mensa.

1. All the secretaries in my office are under 21. All the young ladies in my office are very beautiful. My secretary has long blond hair and blue eyes.

Which statement(s) below can be justified by the information given?

- a) My secretary is under 21.
- b) My secretary is a beautiful young lady.
- c) Neither of the above.
- d) Both of the first two.

2. In a certain field there are both horses and men. There are 26 heads and 82 feet (or hooves) in the field. How many men are there? How many horses?

Here are their answers:

1. The "correct" answer is given as (a) my secretary is under 21. What's supposed to trip you up is the assumption that all secretaries are female, which isn't stated. But what about the assumption that my secretary is in my office? It doesn't say that, but then it doesn't say my secretary is female. Depending on which set of assumptions you choose, all four answers are possible. Although I've never personally had a male secretary, I have worked in situations where my secretary was in another office, even in another city. Does this make me less intelligent than the psychologist who posed the question?
2. By simple algebra, you get the official answer (15 horses and 11 men) if you assume that each horse has 4 feet and each man has 2. I don't know much about horses, but I read this test after watching a Veterans Day parade. It was obvious to me that one possible solution is 16 horses and 10 men, one of whom is a veteran with no legs. Of course, there are lots of other solutions along this line, if you're not a psychologist.

Any intelligent being who has been exposed to this sort of testing has experienced the same frustration: You can think of several possible answers, but you know the psychologists want only one; and you're not allowed to ask questions. It's not so bad when the questions are in a magazine just for fun, or even in the Mensa admission test.

But what if you want to get into college? Or get a job? Or get into a favored track in the second grade? The psychologists hold the power to keep you from getting what you want, and their power is unquestionable.

Quite possibly, academic psychology is the most arrogant profession of all time. In Chapter 1, we already encountered the central dogma of academic psychology: There is one and only one correct solution to every problem—and the psychologist knows it. This dogma applies equally to tests for people or mazes for rats. Any rat who displays a modicum of suspicion for the psychologist's setup runs the maze a little slower and is labeled "less intelligent." To me, anyone who fails to be suspicious of psychologists' experiments should be labeled "less intelligent."

The central dogma is damaging enough to the individual person or rat trapped by the psychologist, but its long-range effects on society may be even worse. Schools and employers reward people whose thinking happens to match the thinking of psychologists, so people either learn to think that way or find themselves out in the cold. After a while, we find people in problem-solving situations who literally believe that every problem has one and only one solution, a solution so inevitable that they will recognize it when they find it.

For a would-be problem-solving leader, belief in the central dogma is 'debilitating disease and the third great obstacle to becoming a star problem-solving leader. Infected designers rarely consider an adequate number of alternative designs, and never consider testing the design other than by their own intuition. Infected programmers are powerless in the face of a bug that deviates in any way from the obvious answer.

Managers infected by the central dogma act like psychologists; they assign work to their subordinates and expect to have it done in one right way—their way. Before long, they create another generation in their own image.

SUMMARY

They're worth summarizing, these three great obstacles to innovation:

1. Self-blindness, concealing your own behavior, so you have no chance of changing
2. No-Problem Syndrome, convincing you that you already know the answer to all problems
3. Belief in the central dogma of academic psychology, blinding you to alternative solutions, even ones you could generate without help from anyone else

These deeply imbedded obstacles form a closed system, standing in the way of their own removal. People who are self-blind might read this list and nod their heads in agreement, but about other people. People who suffer from NPS will not even read the list in the first place. People who believe the central dogma will already be on what they consider to be the one and only path to success.

So let's bid those people goodbye and concentrate on those for whom there is still some hope. Let's stick with you creative people who know the double enjoyment of watching what you're doing while you're doing it and laughing at yourself while you watch. To you, I offer the following chapters on how to overcome some lesser obstacles in your path to becoming a leader through innovation.

QUESTIONS

1. Knowing what you had for dessert is an application of general self-awareness to the question of health. How aware are you of your own health and how it affects your leadership style? How do you feel right now? Take an inventory of your physical condition, and describe how it currently affects your performance as a leader.
2. Poor health is an obstacle to innovation and just about everything else. How has your long-term health affected your career? What is your health going to be like in the future? If you have a hard time answering that question, what makes you think your health is not under your own control? What are you doing to keep it under your own control? How will it affect your career in the future?

3. In answering the previous two questions, did you respond that your health was "no problem"? What does that tell you about yourself?
4. Do you know your IQ? Do you let other people know? Does knowing your IQ affect your ability to lead? How?
5. Do you like to take tests? If you knew you were assured of doing well, would you like to take a test? What if you were assured of doing poorly? What if you had to take the test, but were never to find out how well you did? What do these questions have to do with leadership style?
6. Find some multiple choice quiz and go through it in the following way: Instead of picking one answer, take each answer in turn and give a good reason why that could be the answer. Then give a good reason for some answer that isn't among the choices.
7. Next time you're in a meeting and several ideas are brought up, apply the technique of the previous question. That is, make sure you give a good reason to the meeting's participants to explain why each idea could be the solution you're seeking. Then offer at least one more...



So who all wished us on our 4th Birthday?
Watch this video to find out.

Biography

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



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

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

To know more about Gerald and his work, please visit his Official Website [here](#) .

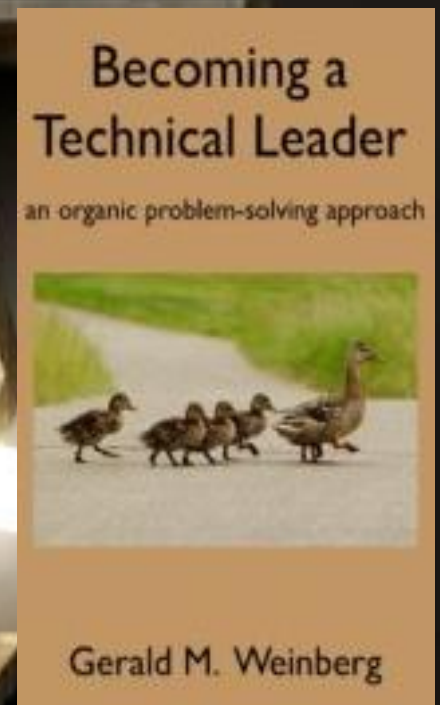
Gerald can be reached at hardpretzel@earthlink.net or on twitter @JerryWeinberg

Becoming a Technical Leader is a personalized guide to developing the qualities that make a successful leader. It identifies which leadership skills are most effective in a technical environment and why technical people have characteristic trouble in making the transition to a leadership role. For anyone who is a leader, hopes to be one, or would like to avoid being one.

This is an excellent book for anyone who is a leader, who wants to be a leader, or who thinks only people with 'leader' or 'manager' in their title are leaders.

Its sample can be [read online](#).

Know more about Jerry's writing on software on [his website](#).



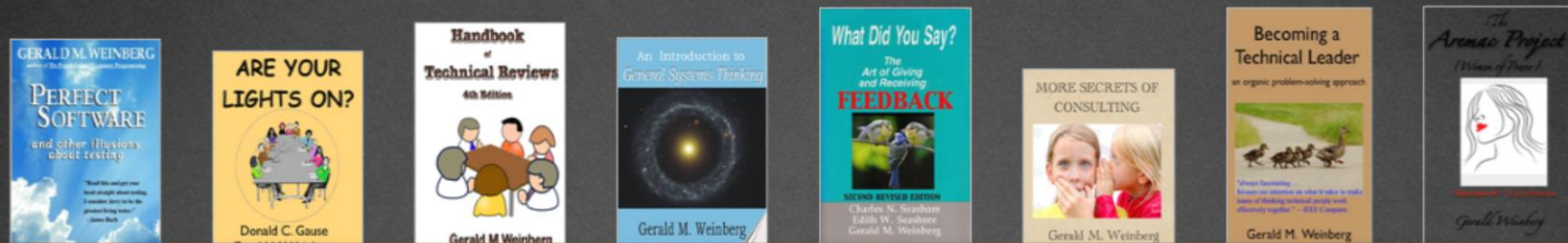
TTWT Rating: ★★★★★

The Bundle of Bliss

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

The Tester's Library

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



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

\$49.99!

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.

Ready! API by SmartBear Software

- by **Lorinda Brandon**

Ready! API is the latest offering from SmartBear Software, a software quality tools vendor that has the most widely used API testing tool in the world, SoapUI. With a focus on what SmartBear refers to as “API Readiness”, Ready! API is a family of tools designed to help organizations ensure API quality end-to-end.

SmartBear is best known as the provider of tools like SoapUI and TestComplete, both mature products that have a loyal user base around the globe. Over the past few years, SmartBear has kept an eye on the growing reliance on APIs throughout the software industry and determined that API quality was one of the most important things to focus on. In 2014, they launched Ready! API, an open core platform built on the framework of SoapUI but offering much more.

What is Ready! API?

The Ready! API family consists of:

SoapUI NG Pro – the next generation (NG) of SoapUI Pro, SmartBear’s commercial API testing tool. With support for REST, SOAP, and a variety of other protocols, SoapUI NG Pro offers the ability to test an API’s functionality by creating complex test suites or simple test cases that perform basic requests against the API. An experienced tester can construct composite projects consisting of multiple dependent test suites, while a tester new to APIs can ramp up fairly quickly with some basic requests and assertions.

LoadUI NG Pro – the next generation (NG) of LoadUI Pro, SmartBear’s API load testing tool. LoadUI NG Pro ingests a SoapUI NG Pro test and allows a tester to construct load scenarios using the same test cases they already defined. With a set of pre-configured load scenarios, a tester can implement a load test within minutes and see real-time test results while the test is running.

ServiceV Pro – this is a new offering from SmartBear that allows a developer or tester to create a virtual API with just a few clicks. Virtual APIs provide much more flexible for testing teams by allowing them to work with an API while it is still under development or is otherwise unavailable. Application testers can also use virtual APIs to simulate API failures or latency then see how their application reacts to those conditions. Combined with the optional (and separately priced) VirtServer, ServiceV Pro not only allows you to create virtual APIs but also share them with internal and external teams.

Secure – this tool is provided free of charge with the purchase of any other tool in the family. The Secure tool includes a variety of security scans designed to find vulnerabilities in the APIs you rely on.

How Can You Benefit From Ready! API?

If your organization has any reliance on APIs, whether they are built internally or you are consuming them from a third party, you should be testing those APIs directly. Ready! API can be a good option for you if:

- You need to perform more than one type of testing (load test, functional test, security test)
- You have multiple organizations that rely on your API for their testing and virtual APIs can help them be more efficient
- You are concerned about ramp-up time. With its easy to use interface and preconfigured load scenarios and security scans, it's simple to get started and be productive with this set of tools.
- You are looking for an affordable alternative to the high-priced enterprise suites that are currently on the market.
- You have a variety of skill sets on your team. With its more advanced features, Ready! API is a powerful tool for the experienced API tester who wants to dive deep into the API itself. But with its sample projects and pre-configurations, Ready! API is an easy tool to get started with for the more junior tester.

Trying Ready! API

SmartBear Software provides free trials for all of their products and Ready! API is no exception. You can download a free 14-day trial of any of the tools in the Ready! API family [here](#).



A green, teardrop-shaped pendulum bob hangs from a thin wire. Below it, on a surface of light-colored sand, is a circular pattern drawn in the sand, resembling a stylized flower or a complex geometric design. The entire scene is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk

LinkedIn PDCA

Are you ready?

- by Rajesh Mathur

Since I moved to Australia few months ago, I receive a lot of requests from fellow testers from either within Australia or from other countries to assist them find a job.

These days I receive 5 to 10 emails on average every week for connecting over LinkedIn and asking me if I had any suitable job/s available for the requester. I almost always respond. If there is something to offer, I do; or if I know someone who can help, I direct people in that direction. If there is nothing I can help with, I offer my advice. I look at people's profiles and try helping them out. If they are not ready, I say it!

There is nothing wrong in asking for help or connecting with others. In my opinion, it is indeed good or OK to connect with potential employers and/or people in one's industry if one is either considering a job change or wants to expand one's network. As they say, "It's not always about what you know; it's also about who you know." However, what is not OK is to not prepare yourself for that interaction. If you have not done your homework and if you are not lucky enough, your chances of losing that opportunity are very high.

See, the first impression is not entirely the last impression, but if not managed well, it may potentially ruin your chances of getting any further traction or communication from the people you are seeking support. Do not send the request just because you can or just for the sake of it. Do not let the chances of getting a job diminish.

So, before you send that email or LinkedIn message, sit back, relax, take a deep breath and follow this advice:

- 1. Know your audience:** Have a thorough look at the receiver's profile. Try to find what they are looking for, what their interests are, who they follow, what kind of skill they possess or admire and who they have hired if you can find the job description. If they are on Twitter, look at their tweets and see what they are tweeting about. This will give you a fair idea about them. It is not too hard to find about people's professional interests on LinkedIn.
- 2. Stand out:** If your profile looks like a million others, you have very little chance to be noticed. Make sure your profile stands out from the crowd. It is not just employers; it is also recruiters who receive CVs with exactly the same skill set and experience. So, stop saying that you are ISTQB certified, know manual and automated testing, can write test plan and test strategy, can do functional, system, regression, integration, blah blah blah testing etc. No employer or recruiter will pay attention to this because you suddenly become part of the crowd.
- 3. Certification show off:** If you are relying on your ISTQB certification and automation certification, good luck to you. Everyone seems to have those these days. You are not special. If you do not have a degree, that is fine. I am more than happy to speak to you if you fulfill the skill criteria that I have mentioned below. If you do not have a testing certificate, that is more than fine. If you do have a testing certificate, beware, I am not a fan of testing certificates and you will have to prove what value you received from those.

Not having a degree is not a restriction. Self-education has its own benefits. I would suggest you read James Bach's "Secrets of a Buccaneer-Scholar: Self-Education and the Pursuit of Passion"

If you are worried that not possessing 'that' testing certification will reduce your chances of being shortlisted, read this piece from Michael Bolton.

Why choose me: 'You' need to prove why you are the right candidate for 'a' role. Again, do not start saying what you say on your CV (look at point # 2). Help employers distinguish you from the entire crowd.

Make it visible: I mentioned this in few of my recent emails to people, "It helps if you have evidence of your contribution to the testing community. So in case you have attended or spoken at conferences, written articles or blogs or follow blogs and forums, mention them in your profile. If you have not done so yet, maybe it is time to do so."

Do it if you really mean it: Send the request if you really mean it and if you are ready to prove that you are the person employers must be looking for or the one who can add tremendous value. This also means that you have done the homework and have good preparation for this interaction.

Search the internet for behavioural interviews if you are not aware of what they are. If you know about those, prepare more.

I have been hiring recently. If you would like to be considered for any upcoming roles, I have listed the basic requirements for testing roles (tester, lead, manager,..) below:

1. Critical, lateral and analytical thinking ability
2. Strong deductive reasoning, attention to detail, persistence, patience and creativity
3. Passion for testing and problem solving
4. Curiosity and questioning skills

What increases your chances to get better attention of your profile:

1. Understanding of Context-Driven testing and strong knowledge of Exploratory testing and reporting styles
2. Strong knowledge of building test ideas using mind maps or similar visualization tools
3. Strong knowledge of bug investigation and reporting skills

You will do much better if you have the skills below:

1. Knowledge of Session Based Test Management
2. Knowledge of any coding or scripting languages
3. RST and/or BBST trainings

None of the above is mandatory. If you think you are a great tester, let's talk! If you want to learn more about practical application of context-driven testing, let's talk!

For further assistance and advice about managing your job search, you may consider reading **this book** by Johanna Rothman.

If you are not aware of what PDCA means, it is Plan, Do, Check, Act.



Rajesh is based in Melbourne, Australia and working as Head of Testing at Australian Health Practitioner Regulation Agency. An International speaker and writer, Rajesh is a context-driven tester who believes in practical approaches of software testing that provide value to the business and stakeholders.

Rajesh has held senior test management positions with Cathay Pacific Airways, Nokia and Sopra-Steria Group amongst others living and working in US, UK, India and Hong Kong China. He can be approached at LinkedIn and on Twitter @rajeshmathur.

[Back To Index](#)



A photograph of several young students in a classroom, seen from behind, with their hands raised in the air. They are facing a chalkboard that has some faint writing on it. The students are wearing colorful shirts: light blue, red, orange, and green. The entire image is framed by a thick black border.

In the school of Testing

for your better learning & sharing experience

Pair Testing: Doing it in Agile projects



Image credit: www.agilebuddha.com

- by Mrs. Rama Komarabathini

Scrum, Kanban, Extreme Programming, Sprints, Standup calls, User stories, Backlogs, Retrospective meetings – do those terms ring a bell?

You may think I must be stupid – “yes of course, they are all Agile related terms” you would say!!

Software projects, especially the product development teams, today are fast moving away from the traditional development methodology and adopting Agile for obvious advantages that it brings to the table. However, as always, advantages are accompanied by a number of challenges.

I have been working as a Lead Tester on Agile projects for quite some time now and trust me, it's a not an envious job. I have seen the challenges faced by testers very closely and I can bet that nobody would like to be in their shoes.

So, what are the practical challenges I have experienced then?

- Agile projects, by characteristic, introduce time-boxed development – hence clearly time is the most important constraint imposed on the testers.
- Agile methodology brings in faster pace of development thereby squeezing the QA team's ability to develop and maintain test cases.
- Agile methodology is based on a key principle of "welcome changing requirements, even late in development cycle". It's a constant struggle for the test team to understand the correct behaviour and ensure test scripts are kept consistent with ever changing requirement.
- Development spill overs - Haven't all of us heard of this complaint by the testers? Yes, it's quite common that the development team over commits thereby further squeezing the time available for the testers.
- Insufficient unit testing – Developers by nature are egoistic lot, or at least that's my perception (apologies if I am hurting the development fraternity here). They feel it is not their job to test the code. They think – "What are the testers for"?
- New features, in every subsequent iteration, churn out code to such an extent that it increases the risk of regression. Now, where would the testers find time to do quality regression testing?

So what am I trying to say here then? Yes, you probably guessed it right – The time available for testing is limited; use whatever is available efficiently and effectively.

The power of two

It's not for nothing that people from time immemorial have talked about the power of two people working together. The well-known proverb in English states it all – "Two heads is better than one".

The notion of two being better than one is also expressed in the New International Version (NIV) of the Bible. The two verses from 9 to 10 of the Chapter 4 from the book Ecclesiastes (Ecclesiastes 4:9-10) quotes the following,

9. Two are better than one, because they have a good return for their work

10. If one falls down, his friend can help him up. But pity the man who falls and has no one to help him up!

You may be wondering – Is there something wrong with me? Why all this philosophy? What has Bible got to do with Agile testing techniques?

Well read on...

The other day, I overheard a conversation of two college going girls. Apparently they had to walk quite a distance to go to the nearest bus stop where they could catch a bus which would eventually take them home. One was saying to the other – "You know, when I walk alone to the bus stop, it so boring and appears to take eternity to reach the bus stop. But when we walk together time flies so fast and it appears that we have covered the same distance in a jiffy".

How true indeed!! I am sure everyone has experienced this sort of a thing or something similar.

So what's happening here? In a nutshell, the work (that of walking to the nearby bus stop) is getting done efficiently and effectively when two people are involved.

This got me wondering – Can I not apply the same concept to testing?

I had read about the concepts of Pair Programming and Pair Testing earlier. At that point in time I felt it was too theoretical. But the conversation of the two girls, in conjunction with the challenges I faced in my project, put these concepts into a perspective.

I was determined to start Pair Testing in my project.

Purpose of this article

Whilst Pair Testing is not a new concept, it is surprising that not many testers use or know about it. Given this fact, it is less surprising that very few organizations have encouraged it or adopted it.

There are many articles on the Internet and elsewhere that discuss Pair Testing and its advantages. However, rather than being theoretical, I wanted to share my experience of using Pair Testing through this article.

So what is Pair Testing?

Very simple, there is no rocket science involved at all. It is a technique in which two people test an application at the same computer by continuously exchanging ideas.

The **pilot**, who is in charge of the keyboard and mouse, will be responsible to perform the actual testing tasks, whilst the **co-pilot** analyses, reviews and guides the pilot. The two members involved could take turns to be pilot and co-pilot at alternative instances.

In my experience I have seen that more often than not, the two people involved, bring in different views and approaches resulting in generating more ideas leading to better testing.

Applying Pair Testing

So how do we apply Pair Testing?

My advice is not to overcomplicate (making a process over complicated takes your attention off your primary goal, which in this case is testing) and follow the KISS principle (Keep It Simple Stupid!!!). I am of the firm belief that any process should be simple, straightforward, flexible and easy to use. Following exactly this principle, I have used the following simple steps to a great effect,

- 1. Determine the duration of testing** – Whilst working in pairs has its own advantages, it deprives the members of thinking individually. It may also lead to one being over dependent on the other. So it is better to work in pair in short bursts. So, keep your pair testing session for a short duration – 60 to 90 minutes per session is ideal.
- 2. Identify scope of testing** – Before starting, the two testers should decide which features should be tested during that session. The duration of testing will also help you determine the scope.
- 3. Establish a goal** – Any activity should result in measurable goal. The same applies here too – establish what should be the result of your pair testing activities. Under most circumstances I tell my pair – “Let’s try to break the features identified in the scope by raising as many issues as possible”.

4. **Determine who would be the pilot and the co-pilot** – Small activity, but very important. Also, ensure that the two members rotate the responsibilities of being pilot or co-pilot. Both of them should get equal opportunities.
5. **Execute the tests** – Start executing the tests by exchanging ideas and discussing about the features. However, if you are seated in a common work area you may disturb others whilst doing so. So I recommend that you book a room for yourselves for the duration of the testing.
6. **Stick to the scope** - Ensure that you do not deviate from the scope of your testing. It is quite possible that you may get newer ideas resulting in increased of scope of testing. However, it is very important to make a note of these and plan a separate session – apply Agile Principles!!!

Applicability of Pair Testing

The steps above seem to be more applicable to “execution of the test cases”. Isn’t it? Yes, you guessed it right. So is pair testing applicable to this task alone?

Now, some are of the belief that Pair Testing is applicable only for execution of the test cases. Some say that Pair Testing cannot be applied to execute normal test cases and should be applicable only for exploratory testing.

I disagree. In my opinion and experience you can apply the concept of working in pairs to any QA activity. This includes functional analysis, designing test cases, execution of test cases, exploratory testing and even bug reporting. However, this depends upon the members involved, their level of expertise and experience and mainly how well they get along together.

If you are planning to introduce the concept of pair testing, I advise you to start it for execution of test cases and then extend it to other activities gradually.

Advantages of Pair Testing

Bringing in efficiency in various QA activities like designing the test cases, their execution and bug reporting, is the key to reducing the time. Carrying out these activities in pairs can help do this for the following reasons,

- **Better knowledge** - With two people working together there is a lower chance of functional misunderstanding. Hence time taken to redo the test cases is avoided.
- **Inherent test case reviews** – There may be no need for reviewing the test cases since they would have been written by two minds put together and only after reaching a consensus. Hence the overhead of reviews is avoided.
- **High creativity** – Brainstorming of the requirements between the two members leads to better understanding of the functionality and hence high creativity, idea generation and better test coverage.
- **Increased productivity** – Two people working together limits interruptions leading to better focus, high productivity and hence better testing and reduced time.

- **Improved testing methodology** – Sharing of past testing experiences leading to improved testing methodology and hence reducing the time required for test cycle.
- **Time saving** – Pair testing normally happens in driver navigator mode (pilot/co-pilot mode) resulting in sharing of tasks and responsibilities.
- **Better bug reporting** – All bugs reported are reviewed by a second person. This reduces the time spent in discussing bugs with developers.
- **Effective training technique** – Pair testing helps in training, both, novices and experienced testers thus reducing the time for a separate functional training session.
- **Better coordination** – Testing in pairs generates positive energy within the team with increased coordination.
- **Better reproduction of bugs** – It becomes easier to reproduce tricky bugs because you start seeing patterns when working in pairs.

Factors for successful implementation

The discussions in previous section may lead you to believe that Pair Testing is probably the next best thing after sliced bread. If so, then think again. It is not as easy as it is explained. As always there are challenges. You will need to put appropriate steps in place to overcome these challenges.

Time, practice and adaptation – Don't expect the process to work the first time. It takes time to get it right. There is no prescribed formula for success. You will need to define appropriate processes and adapt them continuously to suit your organization and employees.

Social inclination – The members involved in pair testing must talk to each other openly and should be comfortable with each other. I have noticed that it takes time for this to happen. So allow the member to spend as much time as possible with each other.

No Jealousy/ego factor – The members involved should be open to criticism from each other. They should open to advice and recommendation for the other members. So if you plan for Pair Testing ensure that you have appropriate training sessions and pair members with care.

Team work – The members should work as a team with a combined goal. This I guess goes without saying and is as true with Pair Testing as with anything else.

Don't measure individual performance, at least not in the beginning – One member of the pair may have excelled compared to the other one. There are many reasons for this and not necessarily skills or expertise. You may have to change the pairs to find the best combination.

Challenges with Pair Testing

Now for some disadvantages – I have seen that Pair Testing has limited advantages in the following situation:

1. If the system under test is not complicated and has a limited number of test cases.
2. You have completely automated your testing process.
3. Testing of application that does a number of things in the background (for example Windows Services) or executes long running processes that span hours.
4. All the challenges listed in the previous section quickly turn into drawbacks if you are not able to overcome them.

Conclusion

I hope I have given you sufficient food for thought in case you are thinking of introducing Pair Testing. However, on a closing note I would also like to caution you that Pair Testing is not a magic wand for all your testing problems. It is complementary to the other testing techniques that you may already employ within your organization. Use it wisely and carefully.

Mrs. Rama Komarabathini, PMI-ACP, PMI-PMP, SSGB, CSTE a seasoned professional with 12 years of experience in IT industry having expertise in Software Testing, Agile Practices, Project Management and Process Improvements. She is currently working as Sr.QA Manager with Advanced Business and Healthcare Solutions Ltd. of Advanced Computer Software Group Plc. Mrs. Rama is spearheading and executing the product test strategy and leading the manual testing practices for the organization. Establishing the COE-Testing and championing the Agile practices across the organization. She has been practicing Agile Methodologies since 7 plus years – Scrum, Kanban, Lean, few practices of eXtreme Programming. She is an active volunteer in entire spectrum of flagship events with PMI Bangalore India Chapter and a strong advocate for Project Management.



Master the Essentials of UI Test Automation: Chapter One



- by Jim Holmes

REAL LIFE GUIDELINES THAT DELIVER RESULTS

Congratulations! You're reading the first article in a series that's intended to get you and your teams started on the path to success with your UI test automation projects.

Important note: After its completion, this series will be gathered up, updated/polished and published as an eBook. We'll also have a follow-on webinar to continue the discussion. Interested?

Register to receive the eBook

Register for the follow on webinar on March 25, 10:00 a.m. ET

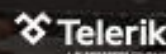
Webinar

Master Test Automation

Dave Haeffner and Jim Holmes

March 25

Register



Chapter One: Introduction

The goal of this series is to help you understand the right questions to ask of you, your team and your organization. There won't be any Best Practices; there won't be any silver bullets. What we hope is to convey the right information to help you get started on the right foot and get through some of the most common problems teams hit when starting out with UI test automation.

This series will walk you through what we think are the most critical aspects of getting a successful, maintainable, *valuable* automation effort in place. The chapters will include:

- **Before You Start:** What are the most critical things you need to think about before starting? We'll walk you through some of the questions to answer as you head off on this journey.
- **People:** You need a great team with specific skills to succeed. We'll help you understand how to build that team.
- **Resources:** Automation requires tools and infrastructure. We'll help you identify things to address as you move forward.
- **Look before You Jump:** Test the assumptions you made during your planning phase by working a prototype, spike or pilot project. Make sure the toolset, skills and process are close to what you need--and adjust.
- **Automation in the Real World:** Now it's time to put things into your real delivery pipeline. You'll read tips for easing your testing process, learn how to really collaborate with developers, and find how backing APIs and creating testable UIs can head off long-term pain.
- **Improving Your Success:** You've headed off on your effort. Now, how can you make it even better and guarantee your long-term success? We'll walk through how to create useful feedback loops that will help you smooth out any rough spots and leverage what's going well.

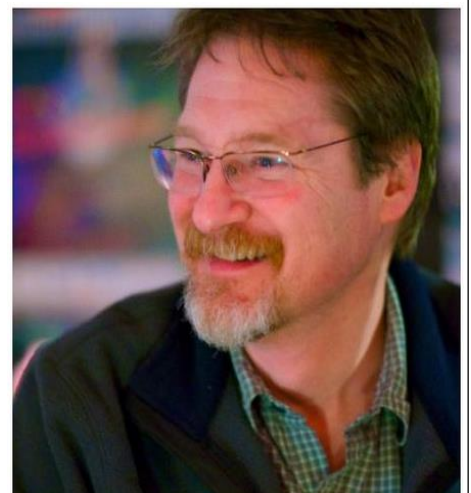
One next page, is a mind map of the chapter/post sequence including some of the topics discussed in each.

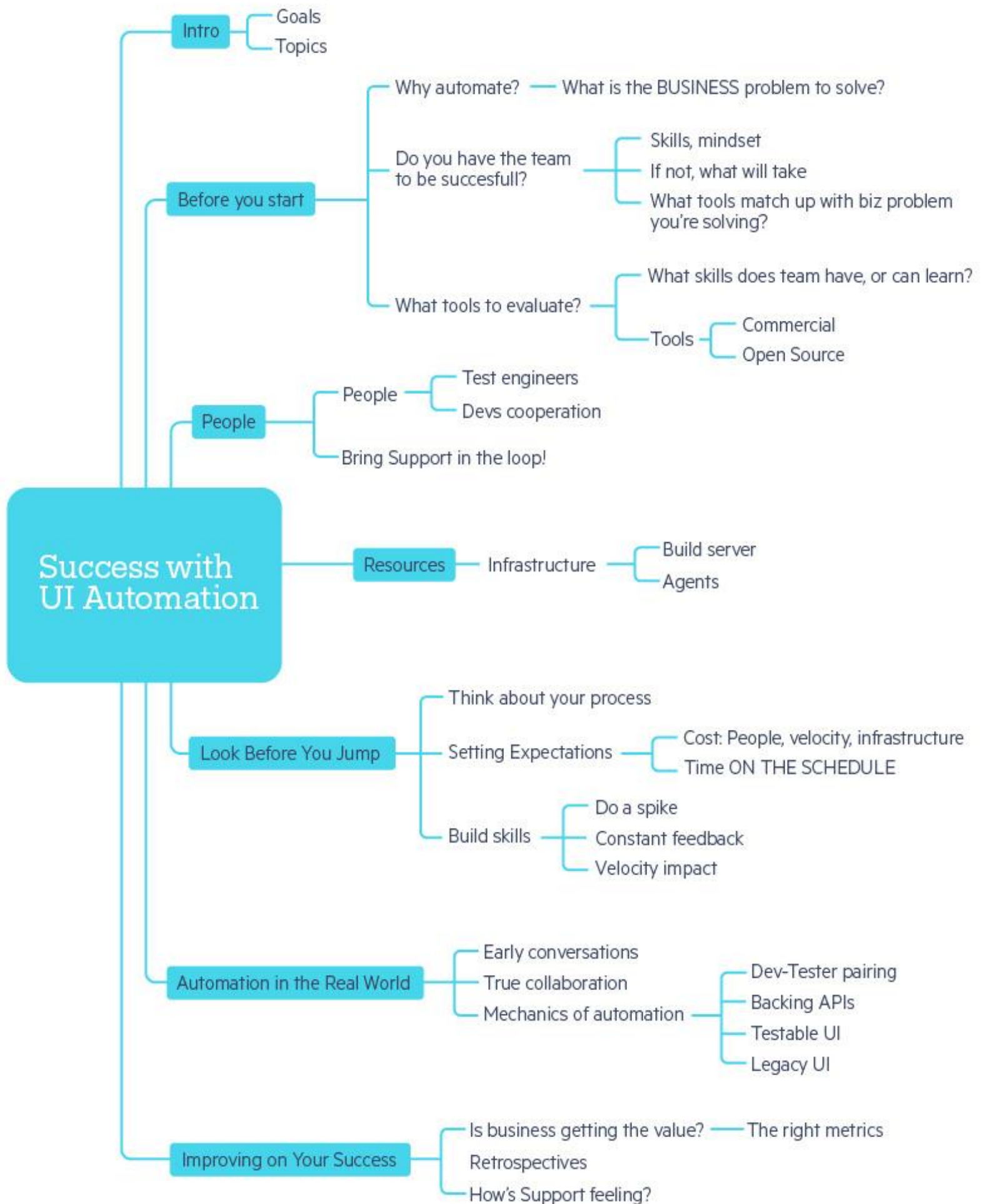
We hope this series will help you plan for your own UI automation projects, or potentially help you identify ways to improve projects on which you're currently working.

Are there specific things you'd like to see in the series? **Let us know.**

Jim Holmes

Jim is the owner/principal of Guidepost Systems. He has been in various corners of the IT world since joining the US Air Force in 1982. He's spent time in LAN/WAN and server management roles in addition to many years helping teams and customers deliver great systems. Jim has worked with organizations ranging from startups to Fortune 100 companies to improve their delivery processes and ship better value to their customers. Jim's been in many different environments but greatly prefers those adopting practices from Lean and Agile communities. When not at work you might find Jim in the kitchen with a glass of wine, playing Xbox, hiking with his family, or banished to the garage while trying to practice his guitar.





Road to Test Automation

- Part 4

- by Georgios Kogketsof

The Road to a Framework

In my previous articles I described how we use the automation-layering model, how decoupling some of the layers lead us towards a framework and what need created the business object layer but we did not explained how these layers are connected with each other.

Let's use as example an application in which data is displayed in a table format and we need to validate the table. As previously explained the table locators could be found in the page object where the locator is defined. On the other hand the methods defining the user actions applied on the table such as 'get table data' are in the business object as a collection of actions defined in the 'page objects common functions'.

When we perform validations of table data we prefer the assertion points to be well defined and usually to have my actual data in the form of a Map (key,value) so my assertions are in the form show in Figure 1:

```
Assert.assertEquals(data.get(key),expected_value)
```

FIGURE 1 ASSERTION

The proposed structure to hold our data is in the form of:

Map1<String,Map2<String,String>>

We selected as keys for Map1 the values retrieved using selenium of the first column and for Map2 keys the names of the columns. For Map2 values should set the values retrieved using selenium of the rest of the columns as shown in Figure 2.

```
Map(column_n_value:Map(column_2_name:column_2_value, ...  
,column_n_name:column_n_value)
```

FIGURE 2 MAP FORMAT

The aforementioned implementation for the key value pairs where chosen to be the table values instead of the table indexes for maintainability purposes.

For example if we construct a map using table indexes, a non-sorted table will bring on row 1 different values every time the page is loaded thus making it impossible to maintain the script. The same situation occurs if a column is added or subtracted, the expected column index will return the wrong value than the expected one. The resulting assertions using table values are shown in Figure 3:

```
Assert.assertEquals(data.get(row_1_value).get(column_2_name),expected_  
value)
```

FIGURE 3 RESULTING ASSERTIONS

For the implementation of the above methodology the first thing we need to do in order to construct the map is to get the number of rows and columns of the table as follows using selenium commands as shown in Figure 4:

```
rows = selenium.getCssCount("css=table tbody tr") .intValue()  
columns = selenium.getCssCount("css=table tbody tr").intValue()
```

FIGURE 4 CODE TO GET TABLE ROWS AND COLUMNS NUMBER

With the number of rows and columns at hand the next step is to retrieve the names of the columns from the table header as follows in groovy as shown in Figure 5:

```

Public List<String> getTableColumnNames(){

    def headerNames = []

    (1..selenium.getCssCount("css=table tbody tr").intValue()).each

        {columns->

            if(selenium.getText("css=table thead tr th" + ":nth-
child("+columns+")").isEmpty()){

                headerNames << selenium.getText(("css=table thead tr
th" + ":nth-child("+columns+")"))

            })

        return headerNames

    }
}

```

FIGURE 5 GET TABLE COLUMN HEADER NAMES

Having the column names the next step is to construct the desired map in groovy as shown in Figure 6 below (Assertion map construction):

```

public HashMap < String, HashMap < String,
String >> getTableInfo ( ) {
    selenium. waitForElement ( componentName ) ;
    def TableMap = [ : ]
    def columnNames = getTableColumnNames ( )
    ( 1 .. selenium . getCssCount ( "css=table tbody
tr" ) . intValue ( ) ) . each { row ->
        def columnMap = [ : ]
        ( 2 .. selenium . getCssCount ( "css=table tbody tr
td" ) . intValue ( ) ) . each { column ->
            columnMap. put ( columnNames [ column - 1 ] ,controller ( ) . getT
ext ( componentName + ":nth-child(" + row + ") *:nth-
child(" + column + ")" ) )
        }
        TableMap. put ( controller ( ) . getText ( componentName + ":nth-
child(" + row + ") td:nth-child(1)" ),columnMap )
    }
    return TableMap ;
}

```

In the above implementation of the table scan an alteration is to accept only td as columns by altering the column map to get:

```
td:nth-child(""+column+"") instead of ,  
*nth-child(""+column+"")
```

The above implementation is a concrete example on how logic can be separated from the page objects and placed in the common layer so every business object can use it as shown in Figure 7:

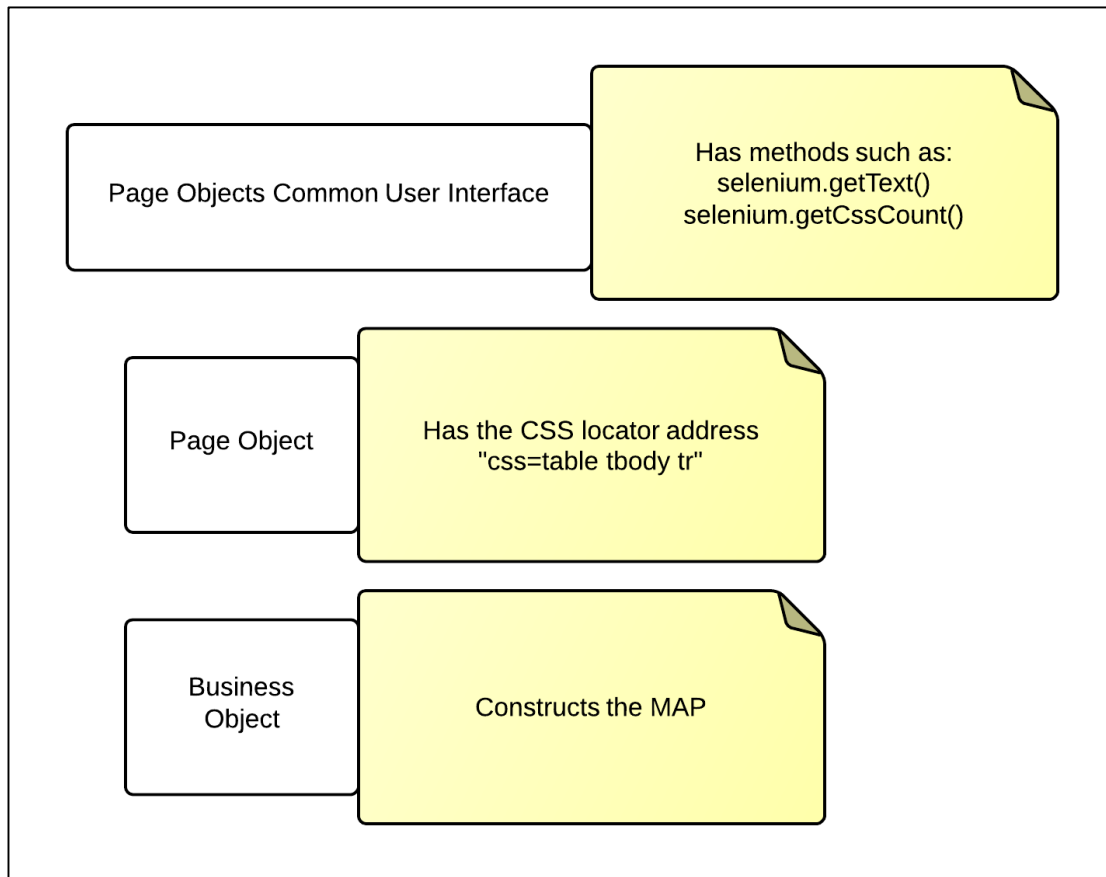


FIGURE 7 LAYERING SEPARATION

An alternative way to accomplish the collection of the table data is to use a dynamic locator setting as variable the name of each row of column one as shown in Figure 8 on next page:

The aforementioned technique is simpler and looping is avoided. It is recommended when tables have a lot of rows but it should be avoided with tables containing large amount of columns.


```

public class MyPageObject {

    public enum CssLocators {

        TABLE ("css= table tbody tr:contains('{0}')),

        /** The my locator. */

        private String myLocator;

        /**

        * Instantiates a new css locators.

        *

        * @param cssLocator the css locator

        */

        CssLocators(String cssLocator) {

            myLocator = cssLocator;

        }

        /**

        * Gets the css.

        *

        * @return the css

        */

        public String getCss() {

            return myLocator;

        }

        /**

        * Gets the with params.

        *

        * @param params the params

        * @return the with params

        */

        public String getWithParams(Object... params) {

            return MessageFormat.format(myLocator, params);

        }

    }

}

```



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>



Sharing is caring! Don't be selfish 😊

[Share](#) this issue with your friends and colleagues!



Other people had a take that was a bit more extreme in the hysterical doomsayer direction and suddenly there seemed to be a large amount of consternation about the wider industry testing actually dying off. It sparked some interesting conversation and debate, but ultimately I don't think anything really changed. Some labels have changed, but testing is still here and shows no signs of shuffling off its mortal coil any time soon. I would like to see procedural, cookie-cutter rote execution style testing die out. I like to think this is slowly occurring. I see less of it these days, but it's still out there.

Things like DevOps, programming Skills for testers seem to be picking up in market. What is your view on coding skills for testers?

I've been talking a bit about the importance of interactional expertise lately – the ability to understand another vocation despite not necessarily having the skills to perform that vocation (think armchair football critic). I think this is a crucial skill (or meta skill) for software testers to cultivate. If you can't code, then at least be able to read and understand code. Be able to talk to programmers about patterns they use and various architecture models. By the same token you want to be able to talk to other non-testing peers in their own terms as well. This helps you not only test code when it's being written, but during design when the solution is being created in the first place. If you can code, you can do more for yourself. Write helper scripts to do tedious things. Create tools to help yourself and others.

I've heard the argument that if testers become too knowledgeable about the inner workings of software, they won't be able to test it objectively, or that testing from a place of ignorance helps you find bugs. This is utter nonsense and it is insulting to both testers and programmers. Testing from a place of ignorance is one type of testing. It can be useful but its one single very narrow track and ignorance is not difficult to find. As far as being objective goes, if I write a piece of code, I may not have the objectivity to effectively test it because of my own expectations and biases of my abilities, how the solution should work and shouldn't fail.

If I hand it to a code-savvy colleague to review, does their understanding of code make them less able or more able to find problems in it?

As far as skills go, coding or otherwise - be a sponge for knowledge. Pick up whatever you can from whomever you can. Add as many strings to your testing bow as you can. Why wouldn't you? Spend time with programmers, with sys-admins, network engineers, UX specialists, BA's whoever you think you can pick up skills from. It's a two-way street though. Think also about how you can add value to what they do.

Back in 2012, you had led a discussion around Coaching Non-testers through Paired Exploratory Testing. Please tell us more about this interesting idea.

These days I think of pairing as a two-way street. Your objective might be to code some piece of functionality, or to test something that has been implemented (sometimes both). It's also an exchange of knowledge and ideas. I think just as programmers can benefit from pairing with non-programmers, testers can also benefit greatly from doing ET with non-testers. I've found it beneficial to both parties, actually. When you're coding, as a tester, you get to see how the guts of the product are put together and have input into how that happens. You can talk to coders about thinking defensively. What might go wrong? What might this affect downstream? What other code or data does this code interact with? What test hooks might we need? What kind of testing beyond the ones we've just written?

The same goes when you're doing exploratory testing. When you test with a programmer, when you start to actually use what they've written, when you start to abuse what they've written it's interesting the sort of ideas that can spark.

Similarly when testing with UX specialists, ideas can snowball around what users might do, what they actually do, how something looks when implemented and operational as opposed to a wireframe or mockup, what you can do to improve the user experience.

Pairing with non-testing peers, whether you're testing or doing other activities, is an excellent way to bring your skills as a tester to bear to add value to the team you're on.

You help organizations hire worthy testers by interviewing them. What is your general observation about hiring/interviews in testing field? What do you particularly look for when you interview testers?

It starts with what the company is looking for. What skills do they need? What do they want this person to add to the company that they don't already have?

I start by looking for a good resume. A well-written resume is very, very rare. So few people seem to know how to put one together. Most of the time it's some flavor of a crap template someone found online. For those who are curious about how to structure a good resume, I recommend Rites of Passage by John Lucht.

In the interview room, I'm looking for someone that knows how to think; someone who is not afraid to question; someone who is passionate and proactive about learning. Someone who wants to know about what they'll be doing, not just someone who wants a job. Skill is important, but often less important than attitude. If you have excellent skills, but a bad attitude, then it's highly unlikely I'll be recommending you.

Remember that not only are you being interviewed, you are interviewing the company. You are agreeing to spend your time – your only finite resource – in exchange for their money. You want to make sure that your time is well spent.

And how can we proceed without talking about Standards/ISO29119 thing? You have had your own share of raising awareness around its side-effect. Would you like share your views on it?

I think slowly, very slowly our non-testing peers are starting to see testing as something more than menial labour and a place that failed programmers go to languish. There seems to be a critical mass of skilled testers that are loud enough to be heard amongst the cacophony of groaning zombies.

ISO29119 is the dying throes of a dinosaur whose time was up decades ago. Command and control test management, blind adherence to templates and procedure, testing by rote – we know these things don't work. We've known for decades. There are enough skilled testers now who are vocal enough and reputable enough to be able to call out this nonsense for what it is. Pick a search engine and look up the standard. The vast majority of hits are serious criticisms of the standard and to date, these criticisms have not been adequately answered; indeed there has been radio silence from the ISO working group for quite some time now. I do not expect that situation to change any time soon.

As a tester, what differentiates you from rest of them?

It's an interesting question. What makes anyone different from anyone else? I look at my circle of friends in the testing community and they are all wonderfully talented people and they put up with me for the most part, so I suppose there are some redeeming qualities in there somewhere.

Our experiences and how we decide what they mean to us are what sets us apart from other individuals. I like to think that some of the different experiences I've had can help me show a situation to other people from multiple points of view. As a tester I think that can be particularly helpful.

To become a next Ben Kelly, one would need to.....

I wouldn't want anyone to become the next me. If there are testers out there aspiring to be anything, they should strive to be the best version of themselves they can possibly be.

Take the person they were yesterday and do something to make that person better. Day by day. Take lessons from others, sure. See what works for you. Do that. Make it better. Make it yours. Emulate, learn the rules then break them. Talk about it. Blog about it. Develop your reputation as a tester of high skill and intellect.

You have mastered the martial arts. Do you think martial arts have helped you get better at testing in any way?

I wouldn't say I've mastered the martial arts, but I have some skill in one particular martial art. I do think having done kendo has certainly informed my testing. I think being able to remain calm under pressure, understanding human nature, recognising patterns and an awareness of your own mental and emotional state are all things I would say I'm better at for having done martial arts. I don't think it necessarily has to be martial arts, but I would recommend that testers try attaining expertise in an unrelated field. You will naturally draw parallels between your job and hobbies. Learning a new thing is an eye-opener especially as you track your own progress through the Dunning Kreuger effect graph.

What metrics do you personally use to manage your testing (and team)? What is your opinion about traditional testing metrics?

Nothing too formal. Right now I'm leading a dev team, so the things I'm looking at are things like the number of stories in play at any given point in time. Are we reviewing and properly testing stuff before moving onto new stories? Is the customer happy with what we've produced for them? Is the end user happy with the service we've provided?

These are not things you can easily add numbers and charts to, though I suppose you could if you wanted. They are qualitative metrics that help us keep our finger on the pulse of what we do. That's appropriate for the work we're doing right now. I dislike command and control management.

I dislike command and control management. I think it provides an easy way for poor managers to blame others for their own lack of skill. I don't think it works well - certainly not for knowledge work. If you don't trust someone to be doing their job to the best of their ability, that's either a conversation you need to have right now, or one of you probably shouldn't be working with the other.

As for traditional testing metrics, it depends on what you use them for. If you are using them to begin a discussion then that may be useful. If you're using them to replace conversation or worse still, track the effectiveness of people then here be dragons. Doing this is a great way to drive dysfunction. If someone's job or pay raise depends on them meeting a metric, they will do that at the expense of all else. Be careful about the sort of behaviour you drive with your measurements.

Metrics are meant to take something complex and simplify it to be more easily understandable. Unfortunately they also have the tendency to take various incompatible things, treat them the same way and obscure important information rather than reveal it. Paul Holland has written [some great stuff about bad metrics](#). I highly recommend reading it.

In coming years, where do you think testing would be? Will it be dead again or will it find a firm ground?

I think with rising awareness of skilled testing by non-testers comes an unwillingness to accept factory-style testing. I think that will only be a good thing for software testing in general. We're a long way from testing being where it needs to be, but at long last I think we are making progress from where we were when I started testing. Seeing the glacial pace of this shift makes me so much more grateful to the testers who have been in this game longer than I have. The work they have done, the energy and time they have spent, is a great gift to all testers who have followed. It is up to those of us who are emerging as testers to take their work and continue to build on it.

Your message to our readers would be.....

You have a voice. Use it! Share your thoughts, your ideas. You can help others and help yourself by participating in the testing community. There are a huge number of talented testers out there and they're accessible. All you have to do is reach out. Stand up and be counted as a tester who values skill and intellect. Help wake up your testing peers (and your managers!) and show them how rewarding testing can be if only they take ownership of their own learning.

You have written articles for TTWt on multiple occasions. What do you think about this magazine? We will appreciate your feedback and suggestions.

This magazine provides a way for testers to learn and grow and connect with other people of like minds. You give a voice to testers who are finding their feet in the testing world, as well as providing insights into how some of the greatest testers in the world think. Keep doing what you're doing.



Happiness is....

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



Like our FACEBOOK page for more of such happiness

<https://www.facebook.com/TtimewidTesters>



YEAR I ~ ISSUE II



SOFTWARE TOOLS MAGAZINE

STATE *of* TESTING



**Thank you for participating.
The report is coming soon!**

T ' Talks



T. Ashok exclusively on software testing

"Friction-less testing" for developers

Very often in discussions with senior technical folks, the topic of developer testing and early stage quality pops up. And it is always about 'we do not do good enough developer testing' and how it has increased post release support. And they are keen on knowing 'how to make developers test better and diligently' and outlining how their solution approach are via automation and stricter process. The philosophy is always about "more early testing" which typically has been harder to implement.

Should we really test harder? Well it is necessary to dig into basics now. Let me share my view as to what they probably mean by testing. My understanding is that they see testing as dynamic evaluation to ascertain correctness. To come up with test cases that will be executed using a tool or a human and check correctness by examining the results. And therefore good developer testing is always about designing test cases and executing them.

And that is where the problem is. Already under immense time pressure, the developer faces serious time crunch to design test cases, execute (possible after automating them). In the case when it does happen, they all pass! (Not that you would know if they fail!). And the reason that I have observed for the 'high pass rate' is that test cases are most often conformance oriented. When non-conforming data hits the system, Oops happens!

So should we continue to test harder? What if we changed our views? (1) That testing need not be limited to dynamic evaluation, but could also be done by via static proving. That is, ascertaining correctness not only via execution of test cases but by thinking through what can happen with the data sets. (2) That instead of commencing evaluation with conformance test cases, we start in the reverse with non-conforming data sets first. Prove that the system rejects bad inputs before we evaluate for conformance correctness. (3) That instead of designing test cases for every entity, we use a potential defect type (PDT) catalog as a base to check for non-conformances first. Using PDT catalog as the base for non-conformance check preferably via static proving and devising entity specific positive data sets for conformance correctness.

So how do these views shift us to do better developer testing at an early stage? Well, the biggest shift is about doing less by being friction-less. To enable smooth evaluation by using PDT catalog to reduce design effort, applying static proving to think better and reduce/prevent defects rather than executing, and finally focusing on issues (i.e. PDTs) first complementing the typical 'constructive mentality' that we as developers have. Rather than do more with stricter process, let us loosen and simplify, to enable 'friction-less evaluation'.

Think & prove vs Execute & evaluate

Picking up a PDT from the catalog and applying a mental model of the entity's behavior can enable us to rapidly find potential holes in implementation. To enable easy implementation of this idea, let us group the PDTs into three levels. The first one deals with incorrect inputs only, while the second one deals with incorrect ways to accept these inputs while the last set deals with potential incorrect internal aspects related to code structure and external environment. Let the act of proving robustness to deal with non-conformances proceed from level 1 through 3 commencing by thinking through (1) what may happen when incorrect inputs are injected (2) how does interface handle incorrect order/relationship of these inputs and finally (3) how entity handles (incorrect) internal aspects of structure like resource allocation, exception handling, multi-way exits, timing/synchronisation or misconfigured/starved external environment.

Non-conformance first

Recently a senior executive was stating that his organisation's policy for developer testing was based on 'minimal acceptance' i.e. ascertain if the entity worked with right inputs. As a result the test cases were more 'positive' and would pass. Post release was a pain, as failure due to basic non-conforming inputs would make the customer very irritated. And the reason cited for the 'minimal acceptance criteria' was the lack of time to test the corner cases. Here the evaluation was primarily done dynamically i.e. executing test cases. When we get into the 'Think & Prove' mode, it makes far better sense to commence with thinking how the entity will handle non-conformance by looking at each error injection and potential fault propagation. As a developer, we are familiar with the code implementation and therefore running the mental model with a PDT is far easier. This provides a good balance to code construction.

PDTs instead of test cases

Commencing with non-conformance is best done by using patterns of non-conformance and this is what a PDT is all about. It is not an exact instantiation of incorrect values be it at any of the levels (1-3), it is rather a set of values satisfying a condition violation. This kind of thinking lends to generalisation and therefore simplifies test design reducing friction and optimising time.

To summarise the goal was to enable to build high quality early stage entity code and we approached this by being 'friction-less'. By changing our views and doing less. By static evaluation rather than resort

to only dynamic evaluation. By focusing on robustness first and then conformance. By using PDT catalog rather than specific test cases.

Once the entity under development has gone through levels 1-3 quickly, it is necessary to come up specific conformance test cases and then dynamically evaluate them if the entity is non-trivial. If the entity under development is not a new one, but one that is being modified, then think through the interactions with other entities and how this may enable propagation of PDTs first before regressing.

So if you want to improve early stage quality, smoothen the surface for developer testing. Make it friction-less. Do less and let the entities shine. It is not doing more testing, it is about being more sensitised and doing less. Let the evaluation by a developer weave naturally and not be another burdensome task.

[Back To Index](#) 



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

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

He can be reached at ash@stagsoftware.com



HOW TO TEST A USER STORY – QUESTION TO UNDERSTAND



Watch this webinar exclusively on www.tvfortesters.com

testing intelligence

- *its all about becoming an intelligent tester*



an exclusive series by **Joel Montvelisky**

Letter to a starting tester

I've been working on the [State of Testing survey](#) and report for the last couple of months, and as part of this project I've talked to a large number of testers and testing teams.

In some of these talks I explained how 17 years ago I started working as an [accidental tester](#), how I tried to escape from testing during the first 3 or 4 years of my career, and how somewhere along the road and without really noticing I found my testing vocation.

After one of these chats I realized that when I was beginning my work as a rookie tester I really had the need for a mentor to help me get started on this journey. There were many times when I would have appreciated professional guidance and advice, especially during some of my moments of doubt.

And so, I decided to write here the email I would have sent to myself back when I started testing to help me cope with some of the main challenges ahead.

Maybe this email is only cheap therapy for myself, but there is also a chance that it may help some of the testers who are only now starting their professional endeavors.

An email to Joel, a starting tester, back in 1998

Dear Joel,

I wanted to send you this mail to help you during some of the difficult times and the challenges that you will encounter as you start your professional career as a tester. What I will tell you may sound lame and trivial at times, but these are the things you will need to hear (and do) to cope with a number of the situations that await you in your coming career.

No one really knows what you need to do better than you do, in the end you will need to figure it out on your own.

Many times you will feel that you don't understand what's expected from you as a tester.

People want you to find the bugs and test the product, but they don't have time to explain what the product really does and how, they will not be open to criticism, they will also hate it when you bring them bad news, and on top of everything else they also expect you to complete all your work within a couple of minutes...

Even though you did not go through any special training, you are suddenly the expert in testing, and sometimes this new responsibility will weigh too much in your shoulders.

As strange as it sounds, no one in your team knows how to do your work better than you do (this will be especially true in the start-up companies where you will work at the beginning of your career). It will be up to you to learn and figure out your job, and to define your tasks in the best way you can with the resources at your disposal.

Don't count on the knowledge of others in your team to rescue you from your responsibility...

There are other testers out there that you can talk too, look for them and share your knowledge and questions.

You are not alone!

Even if you are the only tester in your company, there are still other companies nearby where you will find other testers.

One of the biggest and most important things you will do is to lose your "public embarrassment" and reach out to other testers, to talk with them about your questions, challenges and dilemmas!

You will be amazed about how similar your issues are, and how much you can learn by simply talking to them and coming up with shared ideas on how to solve your professional predicaments.

This will help you learn that asking questions is not a sign of being weak or dumb, but a sign of being professionally confident and smart...

Testing is as much about learning and asking questions, as it is about pressing buttons and reporting bugs.

Hand-in-hand with the last point, you should also learn to ask the people on your team questions about your product, this will help you become a better tester.

Many times you'll see that a developer comes to you to explain a new feature or a change, and after he/she is done explaining (or at least after they think they are done explaining) you have more questions than the ones you started with. When this happens don't be ashamed to ask more questions and to request this person to explain the point from a different angle or using different examples.

A number of (nice) developers may forget that you are not aware of all the technical details, or they will start their explanations based on other assumptions that are not known to you, or they are simply bad communicators and so explain stuff in the worst possible way...

This is just the way it is, and you don't need to be afraid to keep asking until things are clear enough for you to do your work.

It is as much their jobs to make sure you understand how to test as it is to write the code correctly.

Whenever you get a feeling that something is not right, don't keep quiet! Understand what bothers you and communicate this to the team.

At times you will get a feeling that something is not right with your product or your process, and it will be your instinct to think that it is you who made a mistake during your tests.

This will surely be the case many times, but once you have re-checked your assumptions and your procedures, and if you still have that feeling of something not being right make sure to communicate this to others.

Stand your ground when you think you are right.

Sometimes it may be a matter of interpretation, you think that the feature should behave "this way" but the developer thinks that it should behave "that way".

When this happens go to someone else who will help you make the correct choice. Try someone who knows the customer and will be able to provide feedback based on their knowledge of how the users work.

The same goes for the times when you see your project is going to be delayed, but you see people behaving like everything was normal and OK.

If you see that features are slipping, and that the quality of the deliverables is below the status you expected them to be at this time make sure to raise a flag and wave it for the whole team to see it.

After all it is your job to ensure the quality of the process and not only of the product under test!

You are not the gatekeeper of your product by brute force!

Having said all these, it is not your job to stop the bugs (or the versions containing them) from walking out the door.

Your job is to provide visibility into the status of your product and your project, and give everyone on the team the information they need to make their decisions.

After you have given everyone the correct information they will need to make the choice whether to release the product into the field or not. You may be part of this team making the decision, but your voice will never be the only one that counts!

Remember that you may not have all the information related to marketing, sales, competitors, or a range of other factors that are usually involved in the process of deciding whether to release a version to the field or not.

Have fun...

Testing should not be 100% serious all the time!

It is OK to have fun and to joke around with the people in your company, just remember that there are times for joking and there are times for keeping serious.

And one last thing!

When you are working in the Silicon Valley around 1999, look for a company called Google, and ask them if they need a good tester. Even if they pay you only in stock take the job... It will be worth it!



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

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

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

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

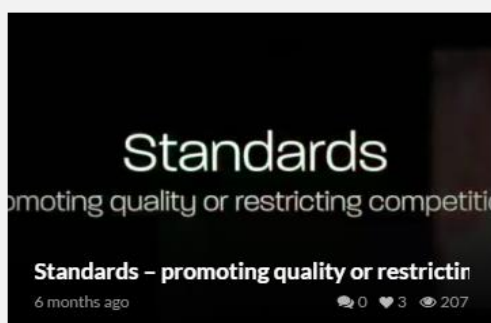
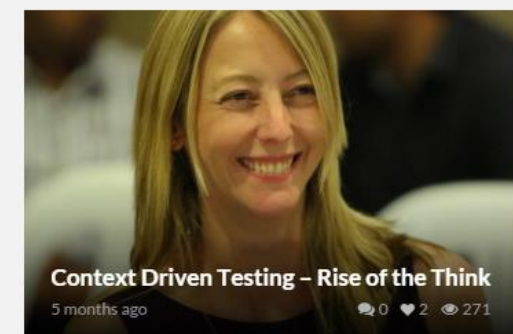
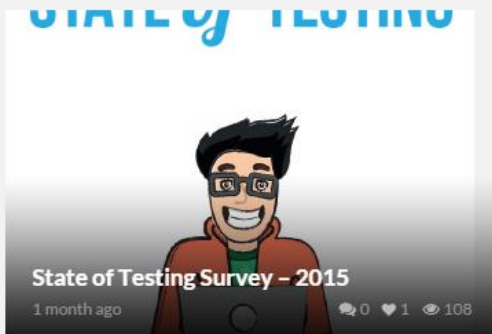
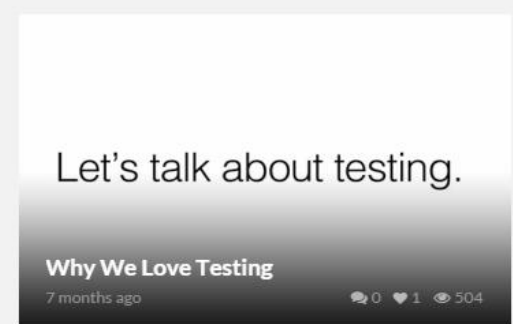
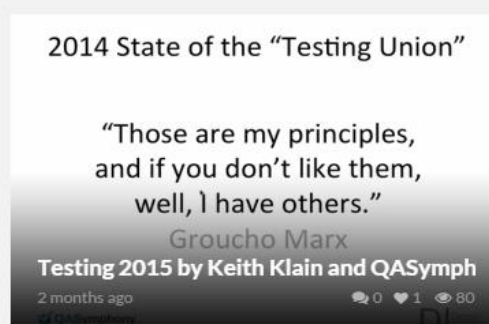
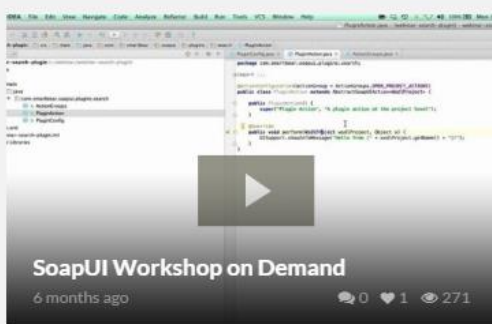
Joel publishes a blog under - <http://qablog.practitest.com> and regularly




Got tired of reading? No problem! Start watching awesome testing videos...

TV for Testers

Your one stop shop for all software testing videos



WWW.TVFORTESTERS.COM



www.talesoftesting.com

What does it take to produce monthly issues of a most read testing magazine?
What makes those interviews and articles a special choice of our editor?
Some stories are not often talked about...otherwise....! Visit to find out about
everything that makes you curious about **Tea-time with Testers!**

Advertise with us

Connect with the audience that MATTER!



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

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

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

Well, hear directly from them.

And the **Good News** is...

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

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





Every Tester

who reads **Tea-time with Testers**,

Recommends it to friends and
colleagues .

What About You ?

in ne>xt issue

articles by -

Jerry Weinberg

T Ashok

Joel Montvelisky

Georgios Kogketsof

...and others

our family

Founder & Editor:

Lalitkumar Bhamare (Pune, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

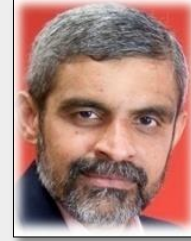
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover page image – Dancing & Dining

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 a **FREE** copy,
Subscribe to our group at



Join our community on



Follow us on - @TtimewidTesters



www.teatimewithtesters.com

