

Tea-time with Testers

NOVEMBER 2011 | YEAR 1 ISSUE X

Jerry Weinberg
Measuring Cost and Value

Darren McMillan
Mind Mapping 101

Anurag Khode
Designing Test Cases for Mobile Applications

T Ashok
Form and Structure MATTERS !



Ben Kelly
What do you mean you don't know
how many were going to St.Ives?

Bernice Ruhland
Onboarding New Testers

Joel Montvelisky
Switching to Agile Testing



TEA-TIME WITH TESTERS

First Indian Testing Magazine to reach 82 Countries in the world !

Created and Published by:

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

Editorial and Advertising Enquiries:

Email: teatimewithtesters@gmail.com
Pratik: (+91) 9819013139
Lalit: (+91) 9960556841

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 in this ezine do not necessarily
reflect those of the editors of ***Tea-time with Testers.***



Dear Readers,

With couple of updates, some good news and yet another line up of great articles, am pleased to offer November issue of Tea-time with Testers.

Update is that we are getting increasing response on our “Teach-Testing” campaign. We have received many letters supporting the idea. I would like to specially thank *Dr. Cem Kaner* for his insightful feedback on this campaign and guiding us through his rich experience. Do not miss his expert analysis that we have published in our *Expressions* section.

Good news is that we have now become the first Indian testing magazine to reach 82 countries in the world! Isn't that exciting ? I thank you all from bottom of my heart for your love and growing support.

Well, there is something more which I personally find more exciting to share. Any guesses ? 😊

We are pleased (rather humbled) to introduce *Mr. Jerry Weinberg*, *Mr. T Ashok* and *Mr. Joel Montvelisky* as an integral part of Tea-time with Testers family. We are thankful to them for considering us worth associating. Their guidance and contribution will always make each of our issue; a priceless one.”

Alright then! I won't take more of your time as you must be eager to scroll down and enjoy the feast 😊 .

Happy Reading ! Have a nice Tea-time !

Yours Sincerely,

Lalitkumar Bhamare



QuickLook



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

Trading Money for Time - 17

What do you mean you don't know how many were going to St. Ives? -23

In the School of Testing

Mind Mapping 101 -31

Onboarding New Testers- 35

Testing Mobile Applications - 41

Testing Intelligence: Switching to Agile Testing - 46

T' Talks

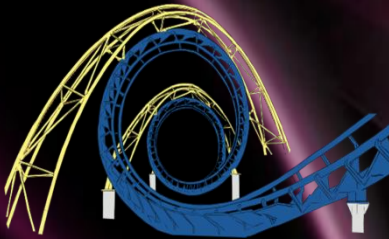
Form and Structure MATTERS! A roller coaster ride experience - 51

Tool Watch

Testing Puzzle – S.T.O.M. Contest

Our Testimonials

Family de Tea-time with Testers



Testing Puzzles

by Sebi



Crossword
by





October 2011

You don't present yourselves as having all the answers, but many of the questions.

Here are a few comments, written while waiting for takeoff on my trip to the AYE Conference.

1. I really identify with Matt Heusser's career story. His editorial is something every young tester (and quite a few older ones) should read. He said he wanted to become a writer, and though he thinks he didn't, he really did.

2. James Bach's anthropological investigation of one test team is a stunning example of what some of us need to do if we are to advance our profession. His method is what anthropologists call "participant observation." I have used this method myself, starting by working with my wife, Dani, the anthropologist as she did her doctoral fieldwork in a peasant village in the Alps. Since then, she and I spent years doing participant observation in software development organizations, which is pretty much how I have learned whatever I know about the business.

I consider it an honor to have worked with James, and shared many hours discussing what we've learned--and more important, how we have learned it.

I strongly suggest that all readers of TTWT begin to learn about how to observe software organizations, and particularly their test practices. Perhaps the most important of the many lessons in James's article is how seriously you can be misled by people's verbal accounts of what they're doing. Watch them. Work with them. And most of all, while doing this, try not to distort their process. After you're finished, then, like James, share your findings with them--but refrain from telling them what to do.

One write, in the Communications of the ACM, said of my courses, "All he does is make people aware of things they weren't aware of before." I think it was meant negatively, but I consider it the greatest compliment I've ever received. And, James, I pass that compliment on to you. I wouldn't say that's "all" you do, but it's the most important of those things. Keep it up. I hope more readers learn to imitate you.

3. Michael Larsen's article about trading off time and money should be one of the required the starting points for training new test leads. And a polishing point even for those leads with years of experience. Too, too often, I've watched the twin addictions:

a. "We must cut costs to a minimum."

b. "We must have all the latest, greatest, tools and equipment."

A reading of Michael's article may be a first step towards a cure of either of these addictions--and they are addictions.

I'm happy to see that the article will be continued in the next issue of TTWT. I hope Michael will eventually cover the tradeoffs not just among money and time, but also with RISK thrown in as a third variable.

4. Aside from the beautiful teacher at the beginning (and Darren's beautiful daughter at the end), I enjoyed the rich story of Darren's evolution as a test-case designer. Much to learn here.

5. Personally, I've always been negative about "Bug Hunts," so I approached Joel Montvelisky's article with great trepidation. I came away with a new feeling about bug hunts. "Instead of never do them," I will now say, "They can be done profitably, but only with much more discipline than the average manager is likely to bring to them. Without that discipline, bug hunts are actually dangerous and counterproductive." And before you do another bug hunt, at least read Joel's article. If you can't do what he says, drop the idea.

THE ISSUE AS A WHOLE: I really appreciate the balance of articles. They're not just all about one aspect of testing, yet they all stimulate thinking, which I believe good articles should always do.

You don't present yourselves as having all the answers, but many of the questions.

I like that, a lot.

- **Gerald M. Weinberg**

New concepts, educational ideas and tools

I would like to congratulate 'Tea Time with Testers' on the excellent job they are doing.

While I have had the opportunity to live and work in different parts of the world. It gives me great pride to know that a magazine created in India, is being globally distributed and being read in 80+ countries around the world.

The key areas where 'Tea Time with Testers' excels where others fail, is in bringing an educational and mentoring aspect to the articles, unlike magazines, that focus primarily on news and opinions. Tea time's articles introduce new concepts, educational ideas and tools. Every time you read the magazine you learn and grow.

Great job guys on the magazine and keep it up.

- **Karthik Subramanian**

Common errors in designing university testing courses.

I teach university courses on software testing. I teach the basic course--black box software testing. The courseware for this is available for free at

<http://www.testingeducation.org/BBST>

or through the National Science Digital Library. These materials are FREE. You can study from the slides and videos on your own or form study groups. You don't need a university to work through this material.

I also teach a course on programmer-testing and a course on test tools. My research center also hosts an annual peer conference, the Workshop on Teaching Software Testing.

There are two very common errors in designing university testing courses.

(1) Attempting to cover too much. Typical testing courses try to teach a mix of white box techniques, black box techniques, metrics, management issues, etc. The result is that the students memorize definitions and easy examples. Universities are reluctant to teach courses like this because they have little value for the students, but they don't know how to narrow the courses and decide what/how to teach at a greater depth (which requires much harder practicals and a lot of personal instructor-to-student coaching). Without this, there is no point in teaching the course.

(2) Some schools do focus their course, but at a very theoretical level. You can teach a very interesting course that frames many applied mathematics problems in software testing examples. For students who are studying for a Ph.D. in Computer Science, this is a helpful approach. For practical testers, maybe not.

To develop an effective advocacy for teaching software testing, consider what advice you will give the schools about what testing students should be able to DO at the end of the course. Without this, the schools have insufficient guidance.

- **Dr. Cem Kaner**

One word to describe this mag: Awesome.

Just finished reading the October issue of Tea time with testers and what a great mag it is.

Being a newbie in testing, I never figured the industry to be so dynamic, exciting and full of challenges. Every day is new day on its own.

I really think that this mag will guide me along this new career and help me expand my knowledge of testing.

One word to describe this mag: Awesome.

Keep it up guys.

- **Zharina Francis**

My Voice on Teach-Testing Campaign

"Yes, Software Testing is a vast domain in itself and it should certainly be included as a study and most importantly R&D course in universities"

Following facts would answer WHY ?

According to a Dataquest survey,

1. 60% of organizations reported that 20% of their workforce is dedicated to full-time software testing and this count is only growing.

2. 54% respondents indicated revenue from software testing in the range of \$10-50 mn and 23% indicated this revenue to be in range of 50-100 mn.

- **Samarjeet Mohanti**

Thank you for all your efforts .

I have been reading Tea Time with testers since July 2011. It is really good to see a magazine being circulated worldwide specific to testing. When I started in the testing field, there was a very few individuals who thought to make a career into this area. Seeing testing evolving over the years I am very happy and convinced I took the right decision 10 years back ☺ . Thank you for all your efforts.

- **Krishnan Subramanian**

After reading your magazine, I feel more charged

Tea time with Testers magazine is really awesome. I am a fresher in the software testing field. I chose this field because of my interest in exploring things. But at times, when I heard from my friends who are developers, saying that software testing is a field that has no scope, I felt bad. But after reading your magazine, I feel more charged now and have immense interest towards testing than before. Thanks for that.

My best wishes for your team. Eagerly waiting for your upcoming issues.

- **Radhi Balan**

TO SEND YOUR LETTERS:

WRITE TO US AT –

teatimewithtesters@gmail.com





What's making News?

- find out the latest happenings in the technology world

Malaysia centre for testing oil and gas software

Thursday, November 17, 2011:

Oil and gas software company IDS has set up a testing centre for oil and gas software in Malaysia, working together with the Malaysian Software Testing Board (MSTB), Universiti Malaysia Sarawak (UNIMAS), and Swinburne University of Technology, Sarawak.

The centre is called "the Sarawak Chapter of the Malaysian Software Testing Hub (MSTH)."

It is part of the Malaysian Software Testing Hub initiative, which began in 2009, with government stimulus money, aiming to make Malaysia and South East Asia a 'centre of excellence' for software testing.

"This is a hugely significant step in confirming Malaysia as a centre of digital excellence. IDS will be an integral part of this effort, with initiatives to further develop product innovation and the skill level of software professionals in Sarawak, and Kuching in particular," said Reuben Wee, Chief Technology Officer for IDS.

The signing ceremony was witnessed by YB Datuk Haji Fadillah Yusof, Deputy Minister of Science, Technology and Innovation of Malaysia.

Software Testing using Visual Studio 2010 from Packt Publishing is on Kindle

Packt Publishing is pleased to report that Software Testing using Visual Studio 2010 is available to Kindle users on Amazon's Kindle Platform. The Kindle eBook represents a step by step guide to understanding the features and concepts of testing applications using Microsoft Visual Studio 2010 and is a valuable addition to the bookshelf of Visual Studio professionals.

UK (PRWEB) November 20, 2011:

Birmingham: Packt Publishing is pleased to report that Software Testing using Visual Studio 2010 is available to [Kindle users on Amazon's Kindle Platform](#). The Kindle eBook represents a step by step guide to understanding the features and concepts of testing applications using Microsoft Visual Studio 2010 and is a valuable addition to the bookshelf of Visual Studio professionals.

What is Microsoft Visual Studio? Microsoft Visual Studio is an Integrated Development Environment, from Microsoft, widely used for the development of applications, websites, and web services across a variety of platforms including: Microsoft Windows, the .NET Framework, and Microsoft Silverlight. Software testing represents an important part of the development and implementation process when using VS.

The book starts with a look at different types of tests. It then goes about explaining examples with a step-by-step approach to master concepts and the features needed to help the reader understand testing clearly. Developers, Software testers, and Architects who need to master the range of features offered by the Visual Studio 2010 for testing software applications will find this book to be a worthy addition to their bookshelves.

The book introduces readers to the main types of testing available in Visual Studio for both desktop and web applications, and then walks them through deploying, running, and interpreting the results of tests. Specifics include the utilization of test manager for creating test plans, test suites and requirement based test suites, the creation and customization of code from the action recording, web performance testing, and much more.

[Read more...](#)

For more updates on Software Testing, visit → [Quality Testing - Latest Software Testing News!](#)

Are you interested in publishing the news about your own firm, community, conference etc in Tea-time with Testers?

Feel free to write to us at:

teatimewithtesters@gmail.com with “News Enquiry” in your subject line.

Announcing...

Smart Tester of the Month Awards !!!

❖ Winners for Testing Crossword :

Devender Reddy Pathi , Allocate Soft. PLC ,
London U.K.

Devaganaraja Gopalasetty, Liquid Hub India ,
Hyderabad , India

Sonal Aggarwal , Logic NEXT , Noida , India

❖ Winners for Testing Puzzle :

Nixon Josephraj

Congratulations !

Discussion helps !

How about talking with us on Facebook?

Come ! Let's have a nice Tea-time there !



Find us on
Facebook

[CLICK HERE](#)

Look Who's Rising



Ten Months

10000+ Readers

82 Countries

ONE Magazine

TEA-TIME WITH TESTERS

Subscribe [here](#) Right Away to get our all Issues for FREE

Tea & Testing

with

Jerry Weinberg

Measuring Cost and Value (Part 1)

A software engineering manager from Texas on his first visit to New York City stopped in a Broadway delicatessen for breakfast. He asked the waiter what was a typical New York breakfast. "Lox and bagels," the waiter replied.

"Good," said the Texan, "then I'll have lox and bagels."

After wolfing down his serving, the Texan summoned the waiter.

"That was delicious," he said. "I'll take another order of lox and bagels."

The waiter complied, and again the Texan gobbled down the order. Finally, after consuming four orders of lox and bagels, the waiter arrived with check in hand. "Is there anything else?" he asked.



"Just one more thing," the Texan said, taking the check. "Before I go back to Blue Springs, I need to know. Which is lox and which is bagels?"

Phil Crosby, in *Quality Is Free*, says that the motivation for improving quality always starts with a study of the "cost of quality." (I prefer the term, "value of quality," but it's the same idea.) In my consulting, I frequently talk to managers who seem obsessed with cutting the cost of software or reducing development time, but I seldom find a manager obsessed with improving value. It's easy for them to tell me what it's worth to cut costs or expedite a schedule, but the value of improved quality seems to be something they've never thought of measuring. Perhaps they know which is lox and which is bagels, but they're confused about which is cost and which is value?



1.1 Confusing Cost with Value

In an excellent article on costing systems, Bill Henry begins with this remarkably clear statement about value:

"Why do corporate management and users continue to complain about information systems?

The reason is simple. Information system (IS) managers have not effectively communicated the department's total value to users and to management. As a result, executives have little or no idea how much IS contributes to the company's earnings and to overall business objectives."

Having reached this point, I was burning with anticipation. At last I was going to read an article for IS professionals about the value of their product. I continued eagerly to the third paragraph:

"To resolve this issue, leading companies are establishing a comprehensive IS costing system. At a basic level, costing lets management know exactly what expenses are incurred to produce each "widget" or final product."

In one sentence, the author substitutes "cost" for "value," and value is never heard from again. Although this is an informative article for those wanting to read about costing systems, it says nothing about "value" other than in the title and the first two paragraphs. It perfectly illustrates Oscar Wilde's remark,

"Nowadays, people know the cost of everything and the value of nothing."

When a crunch comes, it often reveals the deep cultural assumptions. When under pressure to justify its existence, an organization has to decide whether to emphasize the cost side or the value side.

Cost counting (project cost, KLOC, function points) is an indication of loss of courage under fire—lack of confidence that what's being done is worth much. Value counting (customer satisfaction, business value, increased sales, and reduced support costs) is an indication that the organization is keeping its perspective—and that its people are confident in their own value.

As De Marco says, "Effort moves to what is counted." Cost counting leads to cost reduction. Value counting leads to value enhancement. Cost reduction is limited by the annual budget. Value enhancement is unlimited.



1.2 What Is Value?

If you're a software professional, you've often heard someone complain,

"Software costs too much."

For the first 10,000 times I heard this remark, I always responded with a question,

"Compared to what?"

Eventually, I tired of that game, which had no visible effect. Now, I reframe the statement and feed it back in the form,

"Do you mean software's not worth enough?"

1.2.1 Perceived value

The value we're talking about is perceived value. Thus, we must know who is doing the perceiving—the group of people we care about in building the system. This also includes those for whom we want the system to have negative value, such as our enemies or competitors.

Here's a perceived value story:

The day after the winner of the \$60,000,000 state lottery was announced; a programmer was reading a dump of his program and noticed that one of the machine instructions, in hexadecimal, was exactly the sequence of digits that won the lottery. He said, "My program was worth half that prize—\$30,000,000—but I didn't know it."

He was, of course, wrong. Yesterday, the winning lottery number was worth \$60,000,000, but the number wasn't perceived in the program. Today it was perceived, but its value was zero. So, its quality was zero both before and after the lottery, but for different reasons.

In many cases, you can test the quality by asking "what's-it-worth?" questions. People would pay a huge sum for a winning lottery number before the lottery, but before the lottery, there's no way of perceiving which ticket will win. That's what makes a lottery interesting.

1.2.2 Collapse of value

The quality question is not always difficult to answer. Sometimes, a software project simply collapses and produces nothing of value at all. There are a number of reasons for such a quality collapse in an organization. All of the reasons interact in a way that no one reason can be called the "cause" of a quality collapse. In other words, a quality collapse is a "systems" problem, in the sense that a system is a collection of things, no one of which can be changed without changing some others. This means that simple-minded solutions like adding large numbers of people will merely make the problems worse.

Underlying all quality collapses, however, is the simple fact that in software, we are attempting to obtain value by achieving higher precision than human beings have ever attempted before. In many software systems, there are more than a billion bits of object code—in some even more—any one of which could be wrong. And one bit wrong—if it happens to be a critical bit—may be quite enough to render the

system valueless even to the most generous customer. In this light, we can say that one bit might have a value of millions or billions of dollars—if it's right.

Furthermore, software is a young, maturing business. Most software systems are relatively new, and have a tendency to alter the way that people think about what a system should do. This means that when we do succeed in producing a high-quality software system, its users want to add function and capacity. This increases the size and complexity of the work. What was an acceptable process for producing quality in the earlier system becomes unacceptable in its larger, more complex successors. Thus, even when software value doesn't collapse, it decays.

1.2.3 The Second Law of Thermodynamics

To err is human; to combat error is software engineering. A great deal of software engineering research has been devoted to eliminating error early, which is all to the good. But no amount of effort will prevent errors from occurring at all—that would violate the two strongest sets of laws we know—of thermodynamics and of human nature.

The Second Law of Thermodynamics says:

To decrease entropy (increase information), you need to add energy.

In colloquial terms, this says:

There's no such thing as a free lunch.

This means that you have to pay to get quality, and the higher quality you want, the more you have to pay. When Phil Crosby says in the title of his book that "quality is free," he doesn't mean you don't have to pay. He means that you'll be more than compensated for what you do pay.

1.2.4 The First Law of Human Nature

Crosby's title, *Quality Is Free*, has great appeal for managers, because of the First Law of Human Nature:

Nobody wants to believe the Second Law of Thermodynamics applies to them.

My job of working with software engineers would have been easier if Crosby had titled his book, *Quality Pays— But Only If You Invest in It*. Although it's not such a catchy title, it tells a more complete version of the quality story, because there is no free lunch. If you want quality, you must pay. Deming once calculated that the cost of quality was 17% of manufacturing cost in a good Western Electric plant. Assuming the Western Electric managers were not stupid enough to spend that much without receiving more in return, then the value of quality must have been much higher.

Unfortunately, paying a high price does not guarantee a fine lunch, just an expensive lunch. The investment in quality must be more than money and hard work. To produce quality consistently, managers must learn new ways of thinking. Simple linear thinking is not adequate to the task of combating the Second Law of Thermodynamics.

Managers will have to become systems thinkers so they can understand the dynamics of quality.

to be continued in Next issue...

[Back To Index](#)



Biography

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



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

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

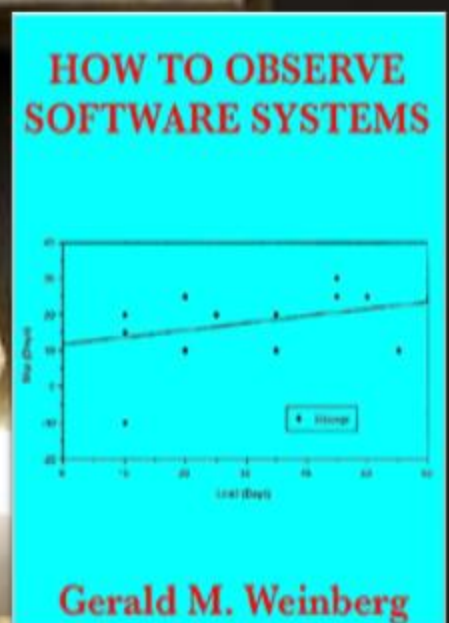
To know more about Gerald and his work, please visit his Official Website [here](#).

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

HOW TO OBSERVE SOFTWARE SYSTEMS is one of the most famous books written by Jerry.

This book will probably make you think twice about some decisions you currently make by reflex. That alone makes it worth reading. "Great to understand the real meaning of non linearity of human based processes and great to highlight how some easy macro indicator can give info about your s/w development process." An incredibly useful book. Its sample can be read online [here](#).

To know more about Jerry's writing on software please click [here](#).



TTWT Rating: ★★★★★

A photograph of a green conical pendulum bob hanging from a thin wire. The bob is positioned over a light-colored sand surface. In the sand, there is a faint, circular mandala-like pattern. The entire image is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk

Trading Money for Time:

When Saving Money Doesn't (And When It Does)- Part 2



By Michael Larsen

What is Time Really Worth?

Let's say an average developer earns \$50/hour. That includes all benefits. \$50/hour is not unreasonable given wages, vacations, holidays, benefits, and sick days. What other aspects of their work day need to be considered? We do not have the ability to simply wake up, teleport ourselves to work, meet our objectives, and then immediately zip away to do other things. We commute and we spend money to make those commutes. When we work at home, we pay for infrastructure to allow that ability. We purchase clothing and food to support us while we are in the mode of working.

Some commentators have said that our full 24 hour day should be viewed in light of what we earn. To really determine the value of our time, we would need to take the total amount of time in a week, and divide it into an hourly figure. If we were to divide those hours in comparison to our average developer earning \$50/hour, that individual is really earning \$11.90/hour.

This is the concept of the "real wage", and it was popularized in the book "Your Money or your Life", written by Vicki Robin, Joe Dominguez, and Monique Tilford.¹ By using this approach, the real wage is almost 80% less than what our mid-level developer earns while working. We can also look at a less extreme example. When all activities and expenses that immediately impact our work are considered, the difference in the real wage can be anywhere from 30%-50% lower than our earned hourly rate. When we realize just how much our real wages are, we feel prompted to do significantly different things with our spending and behavior.

The value of time becomes more critical as additional people come into the picture. Time is multiplied, and each non-optimized hour affects all of the individuals involved in a project. Going back to my previous build server example... imagine a daily build currently takes three hours. Adding another server could bring the time of the build process down to two hours. Put into monetary terms, if a team of 10 developers was able to save just two hours each per week, the team would then have 20 hours more per week to add value to the product. That is a saving of \$1000/week or close to \$50,000/year!

The "real wage" for an organization can be greatly increased by its ability to save time. It can also be greatly decreased by the time it loses.

Is Time Always the Enemy?

It is possible to lose money when time is not taken into account. Time can also be used to an organization's advantage as well. Using the build server example once again, we will need to spend money to get the hardware and to achieve the time savings. What if we could get the same hardware for ½ the cost? Could we get double the hardware for the same cost? For the past 30 years, computer and networking equipment has roughly doubled its performance every 18 months. This doubling capability has also been closely tied to improvements in RAM, disk space, peripherals, networking devices and the cost of internal components of motherboards and expansion cards. We can see a general trend that machines effectively double in power every 18 months to two years, while staying relatively even cost-wise.

With this in mind, does the time not saved, and the subsequent lowering of the real wage of the organization, offset the potential of gaining double the power in machine performance in 18 months? If the answer is "no", then waiting could potentially lower costs while still getting a better return on time. Consider what the cost/benefit analysis of buying a piece of hardware now versus later would be (specifically with the idea of the gained time for our example team providing a \$50,000 cost savings each year).

What other steps could be put off until later? Consider what happens when automating tests. I have a natural tendency to want to get in early and start working with the system as soon as the UI or other elements are coded. By automating tests early, the ability to test the system early and often should yield a good return on the time spent to automate the steps. What happens if the UI changes? Additional fields are added (or removed), or just relocated to a different part of the screen. My automation efforts will have to be modified (best case) or completely redone (worst case). In this instance, is it better to wait until the UI is finished? The odds (and risk) of the UI being changed are now greatly lessened, and my automation efforts are less likely to require reworking or re-doing.

Balancing Short Term and Long Term Gains/Losses

When I look to balance and consider the costs associated with money and time, it's important to look at both short term and long term views. There is always a return on investment when it comes to the time put into any process vs. the value of the output. Whether the return is positive or negative, and by how much, is up to the team and the organization. This may also be dictated by issues that we have little control over. Using automation as an example, there is a formula that Deon Johnson refers to as the Automation Return on Investment (ROI) [2].

The Automation ROI is determined by comparing Cost Savings, Increased Efficiency, and Reduced Risk. The equation we use for this is:

$$\text{ROI} = (\text{Gains} - \text{Investment Costs}) / \text{Investment Costs}$$

The larger challenge is for us to convince management that there is likely to be a positive return for our investment. It may cost our organization \$100,000 to automate a series of tests. If the net result of that automation is quicker time to market and an increase in sales, that's a strong incentive to make the changes. If the \$100,000 spent for automation does not generate additional revenue to equal or surpass the initial investment, it can be argued that the return on investment is negative. In this situation, a manager would be unlikely to recommend continued automation efforts.

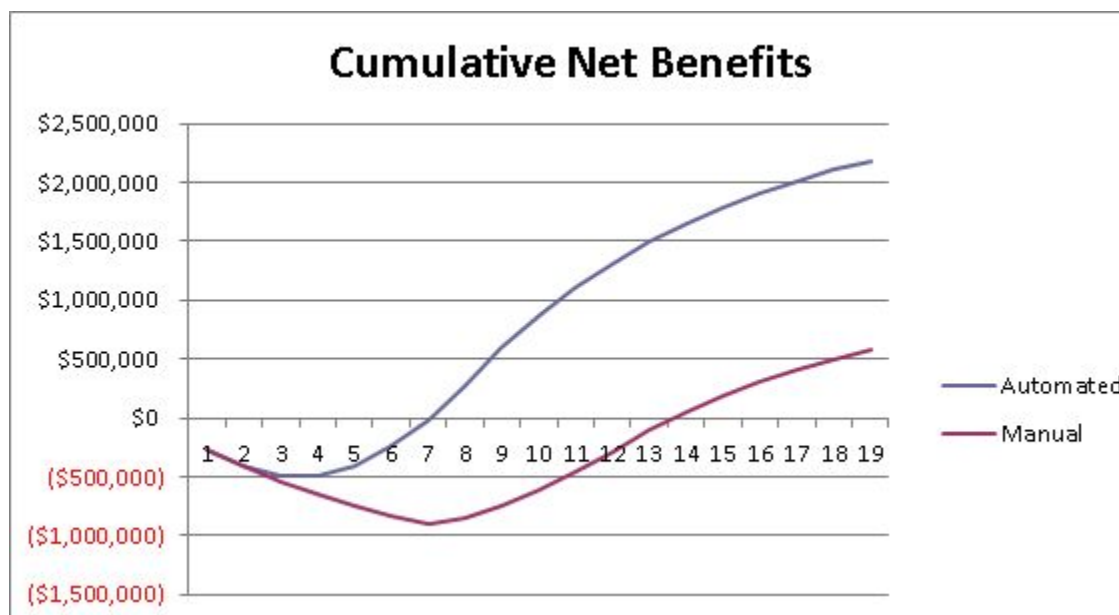


Figure 1: The cumulative benefits over time comparing a range of tests performed manually vs. with some level of automation (numbers are in months) [3].

What I think needs to be considered is the window of time being used. One financial quarter may be far too short a time period to determine the value of a testing effort. Sadly, quarterly numbers are what drive many organizations. In this instance, a \$100,000 deficit may doom a project. However, if a longer view is taken, that initial deficit will be erased. As seen in figure 1, if an investment is made up front, and that investment can get a modest performance increase over an extended period of time, the return on that investment can be considerably higher than not making the investment. Compared to a longer time period (say, three or four years) the benefits and cost savings will become more apparent [4].

As an example of balancing short term expense with long term benefit, Cisco Systems offers an interesting example. They invested time and resources between 1992 and 1995 to create a robust automation framework around the Tcl language. This framework was then used to create extensive automation libraries and script suites to test the Cisco Internetwork Operating System (IOS). Scores of testers were responsible for developing these scripts. These scripts would then communicate on the routers' console ports, and send commands that would set up the router's internal configuration parameters. These Tcl commands, using the Expect extension, were capable of initiating numerous test sequences. The scripts allowed the testers to confirm results, determine the pass/fail criteria and then tear down the tests.

Decades of cumulative man-hours were put into creating the libraries needed to allow the test framework to be completed and enhanced over the initial three years. Large lab facilities were created to house all of the equipment required to run the tests. This required a large up-front expense, but it proved to be an excellent way to run regression tests and to capture information for reports and for

issue tracking. Today, Cisco's investment in Tcl goes beyond testing. Cisco IOS has an extension that allows administrators to use Tcl directly. They can use it on the console port to configure routers and switches in production environments, as well as to create queries for SMTP messages and also for accessing SNMP MIB's [5].

I have also had many experiences where the short term benefits of saving money outweighed the long term issues. One of the favorite "war story" examples used by many testers (and yes, I've experienced my share, too) come from experiences with outsourcing. It's considered a boon when outsourcing effort help you and your team get more done and increase profits. It's seen as a bane when that outsourced team ends up taking over your job. I've been in both situations, and can speak to exactly how both situations feel. The business case I've heard most often used to justify outsourcing testing efforts is that the organization can save money. To be fair, when it is done correctly, with a high level of communication and collaboration, and with a motivated and well-sync'd team, outsourcing can be beneficial and can be a cost saver.

For outsourcing to save money, the entire business case must be considered. For example, I was part of an outsourcing process when I worked with a Japanese video game company. For our group of testers, we were the outsourcing group working on their behalf, helping to ready titles for release and distribution into the U.S. market. We reaped the benefits of that relationship, and provided a good service with a high degree of quality and satisfaction to our clients.

By contrast, a number of years ago, when I was working with a technology company, there was a large initiative where management decided to have an outsourced group assist in the development and the testing. The reasoning was that there was more than enough work for our group to do and this would allow us to quickly get a large number of tests completed and have the reporting turned around to management quickly. That was the goal, a way to get a lot of testing done quickly, and at minimal cost. At first, it looked like the organization would save a lot of money. Over time it became clear that what management had hoped would be tested, and in the timeframe they wanted, wasn't. Several meetings were required to clarify the situation. Each meeting needed to be conducted with the management team in one location and the development and testing team half a world away. This made the effective time between each daily build and receiving results to be two days.

Days turned into weeks and weeks into months. It became clear that any short term cost savings was consumed when the product was not ready to be shipped on time. As the team continued to try and get the product ready, more issues appeared. Some issues were related to usability, some with performance, and quite a few issues affected the overall user experience. Ultimately it was determined that the project would not meet the needs of the customers, and both development and testing were halted. Subsequent work was brought back in house. The cost benefits originally envisioned never materialized. In this case, the cost in both money and time was significant, with a negative return on investment.

So, if outsourcing is an option to consider, here are a few questions we should be asking:

1. What is the cost of trying to collaborate across greatly differing time zones?
2. What are ways that the collaboration can be improved
3. Would tools aid the process of collaboration? If so, how much would they cost?
4. How much travel will be required to keep the two groups in sync?
5. How long will it take to fully ramp up an outsourced team?
6. How will these changes affect the designers of the system?
7. How much does the design team have to document for the outsourced team to be effective in their testing?

8. How much will it cost to produce that documentation?
9. Will there be any long term value in the documentation produced?

Asking these questions up front, and many more, plus considering all costs associated with them, will give a truer picture of the expense, in terms of both money and time.

How Can We Optimize Costs (Money and Time?)

Finding the magic point where we can save money and keep to a good amount of time while ensuring high quality is a delicate balance. Many might say it can't be done. I say with an eye towards having all three perfectly balanced, it's unlikely. I do believe, however, that there is a way to get to where the monetary cost is "low enough", the overall time is "long enough or short enough" and the quality is "good enough" to be effective and release a good product to stakeholders who want it in a timely manner. There is no magical incantation or special formula to let the organization know exactly where that "good enough" is, but there are a number of areas that I feel can be examined and, when taken into context, useful to gauge whether or not a project or application will fulfill the needs of the customer, in a way that is timely and provides for good cost savings to implement [6].

1. Determine the Most Critical Areas to the Stakeholders
2. Focus on the Highest Risk Areas First
3. Make Sure to Ask the Right Questions
4. Fix the Show-Stoppers, and Understand what Constitutes a Show-Stopper
5. The Perfect is the Enemy of the Good

Determine the Most Critical Areas to the Stakeholders

Who is the product for? What is their expectation? Does our solution meet it? If the answer is "yes", then it is safe to say we are on the right path. If it does not, no amount of cost cutting or optimization will matter. When I was sitting in a development and testing meeting, discussing a particularly time sensitive software project, our development manager asked the assembled group: "Do we make software here?!" When the developers and testers said yes, he shot back "No, we do not! We make a solution that solves a problem for our customers. At the end of the day, if we don't get that part right, it doesn't matter what else we did get right!"

Focus on the Highest Risk Areas First

The areas that matter most to our stakeholders are ultimately the highest risk areas. It doesn't matter if we agree or not, they are the ones who want the application, and their wish list is what's driving the purchase of the application. An example might be to make sure that any UI screens can be viewed in an 800x600 window. It may seem like a trivial issue, but it isn't when the target application is going to run on a kiosk where that is the maximum resolution. Screens that do not look good or require scrolling, especially in an application that uses a touch screen or does not have a method for scrolling, will be a major problem. If this is a key consideration, testing with an 800x600 screen better be one of the first priorities in our test plan.

Make Sure to Ask the Right Questions

Who are the intended users? Does a product already exist that meets this need? If we are first to market, does the application do what our customers want it to do? If we are not first to market, what differentiates us from the competition? Is that difference significant enough to make for an appealing alternative with the established players? What kind of environment will the application be deployed in? Does the product live up to the expectations set for it? Are there any legal requirements we need to be aware of? Each of these questions provides valuable information to help testers set their priorities on the right areas. This information helps make sure that, again, the areas that matter the most get the most coverage. Areas that are less important, or not important at all, get prioritized appropriately.

Fix the Show-Stoppers, and Understand What Constitutes a Show-Stopper

Many testers are trained to look at crashes as show stoppers, but they may not understand that a key business rule is being violated, or that a poorly worded paragraph or stray punctuation could prove embarrassing should it get out. If we press a combination of keys, and the application crashes, that looks bad. Many testers will rightfully state that this is a big issue and a “show stopper”, but is it? Is the combination of keys a regular occurrence, such as a combination of regularly used shortcut keys, or is it an unusual combination that is very unlikely to come into everyday use? By comparison, look at the main text that spells out what the users of the software will do as a service for a customer. Imagine that the text is misspelled and the punctuation is wrong. Many testers would consider text issues to be a low level cosmetic bug. That may be true, but in this case, it may be an issue that really sets off an alarm in the customer buying the product. The random key presses that caused the crash was a way lower priority compared to the text on the page that spelled out the agreement between the service provider and their client.

The Perfect is the Enemy of the Good

There may be many small issues that have been found, and there may be many low traffic areas that may not have received thorough testing. The ship window is two weeks away, and we determine that we can cover all of the remaining areas, and many of these little niggling issues can be resolved and retested, but only if we get six weeks of additional testing time. What do we do? When faced with this option, I tend to look at the areas that have not been tested and evaluate the risk if any of those areas were to house a monster bug. What are the odds that area will be hit? What are the chances that the issues I have discovered will be seen as catastrophic? In this case, I rate on a scale of 1 to 5, 1 being really critical and 5 being virtually unlikely to happen, and see what rates a 1, a 5, or in between. I then rank them in order of that gut feeling and I see what I can hit and what I cannot. I may have to leave several 5's on the table, but I better make really sure that I have covered all the 1's in question. After doing that, the project manager, development team and test team, along with any other key stakeholders, will have to make a judgment call. Do we keep digging and testing, to see what else we can uncover, and risk not making our release date? Do we roll with it and decide it's time to let the product go, because we have all decided that, based on the criteria we have received, the risk areas we have covered, the questions we have asked, the answers we have received, the feedback we have received from end users, and the show-stoppers we have fixed, have given us the gut feeling that we are ready to go live? Different situations will inform either of those decisions, but there are times when “good enough” really is, and waiting for perfection will prove more costly than beneficial.

Conclusion

The cost of testing is real. There is always a price we have to pay if we want to improve the quality of the software. Sometimes that cost is in dollars, sometimes that cost in hours, days, weeks or months. Many times, we have to make trade-offs to deal with both. When we look to save money due do not taking additional time into the equation, it's entirely possible that the money saved will be outweighed by the opportunity costs of the time we have lost. Take the time to determine what your "real wage" as a tester, as a manager, or as an organization, actually is. Consider what the value of your time actually is, and make sure to include that when trying to determine how to reduce the cost of testing. With an eye towards looking at both money and time, it's possible to strike a balance where both can provide a positive return on investment.

References

- 1 - Robin, V., Dominguez, J, & Tilford, M. (2008). Your Money or Your Life: 9 Steps to Transforming Your Relationship with Money and Achieving Financial Independence: Revised and Updated for the 21st Century. New York: Penguin.
- 2 - How Is Automation Return-On-Investment (ROI) Calculated?, Automated Testing Institute, http://www.automatedtestinginstitute.com/home/index.php?option=com_content&view=article&id=1097:faq-roi&catid=54:faqs&Itemid=84>, retrieved electronically on August 18, 2010.
- 3 - Scumnotales, J., Why Incremental Development is Better, an ROI Perspective, <http://agile.scumnotales.com/2009/02/why-incremental-development-is-better-an-roi-perspective.html>, retrieved electronically on November 15, 2010
- 4 - Johnson, D, Test Automation ROI, Automated Testing Institute, http://www.automatedtestinginstitute.com/home/articleFiles/articleAndPapers/Test_Automation_ROI.pdf, retrieved electronically on August 18, 2010.
- 5 - Cisco, Cisco IOS Scripting with Tcl, http://www.cisco.com/en/US/docs/ios/12_3t/12_3t2/feature/guide/gt_td.html, retrieved electronically August 18, 2010.
- 6 - Bach, J., Do We Really Need all This Testing?, http://www.quardev.com/content/whitepapers/do_we_really_need_all_this_testing.pdf, retrieved electronically on August 18, 2010.

Discuss this topic on Quality Testing
[Click HERE](#)

[Back To Index](#)



Michael Larsen is a lone tester in San Francisco, California.

He has spent seventeen years in testing and test organizations, ranging from network routers and web caching devices to distributed databases dealing with the legal and entertainment industries.

Michael is a Brown Belt in the Miagi-do School of Software Testing, an instructor with the Association for Software Testing, and the co-founder and facilitator of Weekend Testing Americas.

Michael blogs at www.mkl-testhead.blogspot.com and can be found on twitter at @mkltesthead.



What do you mean you don't know how many were going to St. Ives ?



There is a well known riddle that goes:

As I was going to St. Ives,

I met a man with seven wives.

Every wife had seven sacks.

Every sack had seven cats.

Every cat had seven kits.

Kits, cats, sacks, wives,

How many were going to St. Ives?

The answer is of course 'It depends'.

Were the polygamist and his mobile cat factory coming the opposite way, or did I catch up to them because that many cats in a sack is not something you're going to be moving quickly or quietly?

Assuming for a moment they are heading in the same direction, are there any other destinations before St. Ives that they may be going to? What if some of them are going to St. Ives, but others are going elsewhere?

What if the wives are not there at all? The rhyme didn't say anything specific about meeting them, only the man. We can't provide a specific answer to the question given the information provided.

Why do testers so often begin their answer to questions with 'It depends'? It can be infuriating to people on the receiving end. Just give me a straight answer for a change!

A tester is obliged to maintain healthy skepticism; to reject misplaced certainty. When posed a question of any complexity, all sorts of heuristics begin to fire as the tester analyzes what the question could

mean. Even if it seems there is a very straightforward answer, any tester worth his salt will turn it over and look at it from different angles.

What's not being said?

What **else** could it mean?

If something changes, does the answer also change?

What's the motivation behind the question?

Is it a question that needs reframing?

and so on.

It is probable that the poser of the question wants a simple and straightforward answer. They may in fact be pressuring you subtly or overtly to do just that. I have on numerous occasions been posed a supposedly rhetorical question with an expected answer and have promptly deviated from the script. Some people get annoyed with this. Some people running software projects have a love affair with certainty and precision. I should say the illusion of certainty and precision. When they get upset at your seeming lack of both, it's because it interferes with this illusion.

If they want to put the pressure on, they may say things like 'You're being evasive' or 'You don't really know what you're doing, do you?'.

They may try and cajole you into the same position. 'What's your best guess?', 'If you were sure, what would you say?'

I used to be afraid to say 'I don't know'. I thought it would make me look like I didn't know how to do my job. Of course, I was inexperienced and had a very real fear that I actually didn't know how to do my job. This made all the harder to tell someone I didn't have all the answers. 'I'm new to testing', I thought. 'I don't know the business all that well and these people seem very confident that they know what they're doing'. It was also a pride thing. I felt like I *should* know. All the testing literature I'd read laid out instructions on how to do estimates. I felt stupid that every time I tried, it was an abject failure. I'm not stupid, I thought, I should be able to just reel this information off. They're expecting me to know this stuff.

So I'd give people a specific number that sounded about right when asked for an estimate, or agree that a small change that the product team or a developer wanted to make wouldn't have a big impact on customers. Of course when things took longer than I'd estimated and when something that I had no idea about meant customers were in fact adversely affected, I took a lot of heat because I didn't have the courage to say 'I don't know. Give me some time to find out', or 'I don't know, but it may take somewhere between x and y weeks depending on how well the code is written and what changes between now and then'.

I bowed to the pressure of giving the expected answer when I didn't know. I found that doing that might take the heat off in the short term, but when things didn't pan out as planned, I was the one feeling the heat (when the stakes were much higher).

At that time, I also didn't know enough to reject the notion that the tester is also the gatekeeper of the release decision. I quickly learned that being overly agreeable (if I'm being nice about it - being a 'yes man' if I'm not) as well as the guy where the buck stops for product quality - is a painful place to be. I learned that saying 'I don't know' did not wound my pride, in fact it allowed me to give up the facade of needing to look knowledgeable and replace it with simply being honest. The latter has served me far better than the former ever did.

Your job as a tester is to highlight risk and reveal uncertainty, even when your audience does not want to hear it. It may seem flippant or evasive to give multiple answers, or a non-confirmatory answer to a question, but it is serious business.

Take for example a scenario where a project manager and the CEO come to you for a status update. You know that because of some last minute changes the product team is trying to squeeze in, the project has some serious issues and you've made sure the project manager knows too. You and the project manager get on well and go out for a drink after work occasionally. You don't know the CEO very well, but know she has a reputation for being a ball breaker. You've seen more than one emasculated vendor slinking from her office.

The project manager says to you 'The CEO wants an update on testing.

You guys are on schedule to finish up by the end of the month, right?

Everything's okay?'

At this point, you have a decision to make. The PM has put you in a difficult position. He's looking for a simple 'Yes' from you. You and he both know that if you back him, you will be lying. That might save your PM's job (at least for now), but you'll have ruined your own integrity, shown the PM that you buckle under pressure and transferred the pressure that was on him onto you (given that you said to him that all was well and you did it in the presence of the CEO). With friends like that, who needs enemies?

Conversely, you can reiterate what you told the PM earlier, that as you've already detailed, if the last minute changes can be deferred, then you may be ready by month end. If they stay, then everything is not okay and you won't lie and say otherwise. The situation will no doubt get messy and political, but you will have your integrity intact and both of them will know that your honesty is absolutely not in question. The manner in which you deliver this message is a subject that could fill a series of articles. Suffice it to say it should be crystal clear that you will not lie or in any other way bend the truth.

Now when I'm asked a question, or for my opinion, or for information in general, I try to weigh very carefully what my response will be. An heuristic that has served me well so far is that it's rarely as simple as it first appears. It tends to go hand-in-hand with Jerry Weinberg's 'Rule of three' - If you haven't thought of at least three interpretations of the information you've received, you haven't thought enough about what it might mean.

Having considered a question and prepared some answers, I also find what Jon Bach calls 'safety language' incredibly useful.

Phrases like 'Based on the testing we have done so far, I think that...', 'To the best of my knowledge', 'It appears that', 'The last time we checked, we observed' and so on, allow you to frame information in such a way that you are preserving uncertainty. You are not stating immutable truths, you're giving information based on what you have observed to this point.

This is not a 'cover your arse' strategy (though it can serve this purpose). Rather, it helps you remember that there may well be other information that you haven't considered and you won't have to hastily backpedal upon hearing it. It lets you discuss and explore other possibilities without having to get defensive about your original position. It also allows you to remind people that you are not some omniscient being (and nor is anyone else).

Being the bearer of unwelcome news is seldom a fun thing, nor is refusing to change your story for someone because that would be more convenient for them. It can be difficult to do, but it is important that you keep your integrity even in the face of pressure to do otherwise. If you explain your position and why it is as it is, then they will either respect you for it, or they won't. There's not a lot you can do if your management is corrupt. In the case of the former, then your stock rises in their eyes. In the latter,

you still have your integrity and you can decide at that point if this is someone that you want to be working for.

Fiona Charles has done some excellent work on the subject of speaking truth to power. If you have the opportunity to hear her speak or attend her tutorial on the subject, I would strongly encourage you to do so. In the meantime, take a look at the slides from a presentation she gave at STAReast 2009. Pay particular attention to the sections on leaving a company where management is corrupt and on 'blowing the whistle'.

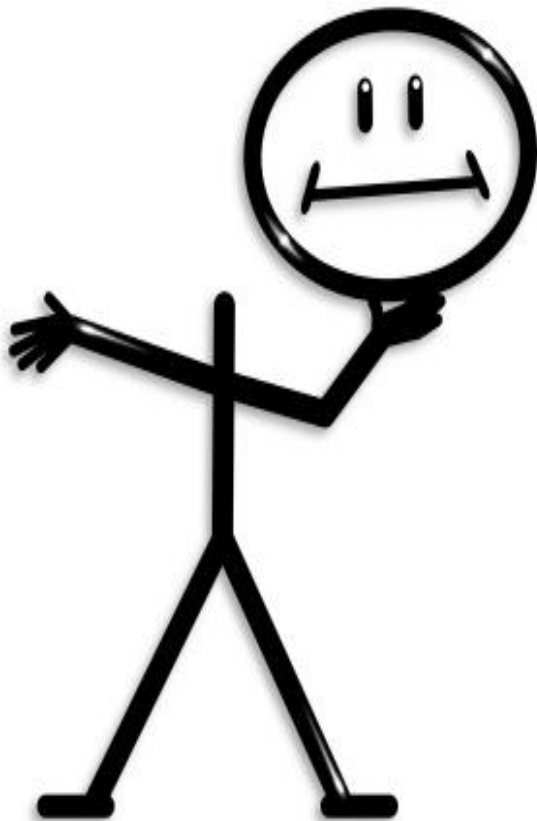
<http://www.choucairtesting.com/portals/2/Documents/StarEast2009/Presentations/W16.pdf>

So what if I say something that my managers take offense to and I get fired? Well, it hasn't happened to me yet, but should it happen, I know that I'll be in good company. There are plenty of testers (Ben Simo and Pradeep Soundararajan to name but two) who have been fired for not compromising their principles. A good tester will land on their feet. They tend not to stay on the market for very long.

[Back To Index](#) 

Ben Kelly is a software tester who lives and works in Tokyo Japan. He has spent the past eight years testing with stints in Internet measurement, insurance and most recently online learning at Cerego, Japan.

When he is not agitating lively debate in the comments of other people's blogs, He sporadically blogs about his own testing experience at testjutsu.com and rants on twitter @benjaminkelly.



I Don't Know.

Plenty of software testing books tell you how to test well; this one tells you how to do it while decreasing your testing budget.

A series of essays written by some of the leading minds in software testing, **How to Reduce the Cost of Software Testing** provides tips, tactics, and techniques to help readers accelerate the testing process, improve the performance of the test teams, and lower costs.

20% OFF for
our readers

How to Reduce the Cost of Software Testing

Edited by Matthew Heusser and Govind Kulkarni

Com Kaner

Matt Heusser

Savina DeLise

Govind Kulkarni

Catherine Powell

Michael Larson

Michael Burton

Ed Barkley

Michael Kelly

Jochen Rosink

Karen Johns

Petteri Lyytinen

David Gilbert

Markus Gartner

Justin Hunter • Gary Beck

Curtis Stachniewski

Scott Barber

Matt Heusser

Catherine Powell

Jonathan Bach

Anne-Marie Charrel

Foreword

What Will This Cost Us?

The Cost of Quality

Testing Economics

Opportunity Cost of Testing

Trading Money for Time

An Analysis of Costs in Software Testing

Test Readiness: Be Ready to Test When the Software Is Ready to Be Tested

Session-Based Test Management

Postpone Costs to Next Release

Cost Reduction through Reusable Test Assets

You Can't Waste Money on a Defect That Isn't There

A Nimble Test Plan: Removing the Cost of Overplanning

Exploiting the Testing Bottleneck

Design of Experiments-Based Test Case Design

Reducing Costs by Increasing Test Craftsmanship

Right-sizing the Cost of Testing

Immediate Strategies to Reduce Test Cost

25 Tips to Reduce Testing Cost Today

Rapid Test Augmentation

Cost of Starting Up a Test Team

 **CRC Press**
Taylor & Francis Group
AN AUBACH BOOK

Enter your Discount Code as

813DA

to avail 20% off on CRC Press.

CLICK HERE to purchase it right away.

Do you have any Questions or Feedback on articles that we publish in Tea-time with Testers?

No Problemo! We will publish your Feedback/Comments and also the answers to your Questions that you have for our Authors.

Do write us your Feedback and Questions in below format and send it to teatimewithtesters@gmail.com :

- Your Name
- Your Brief Introduction
- Article Name
- Your Feedback or Questions if any

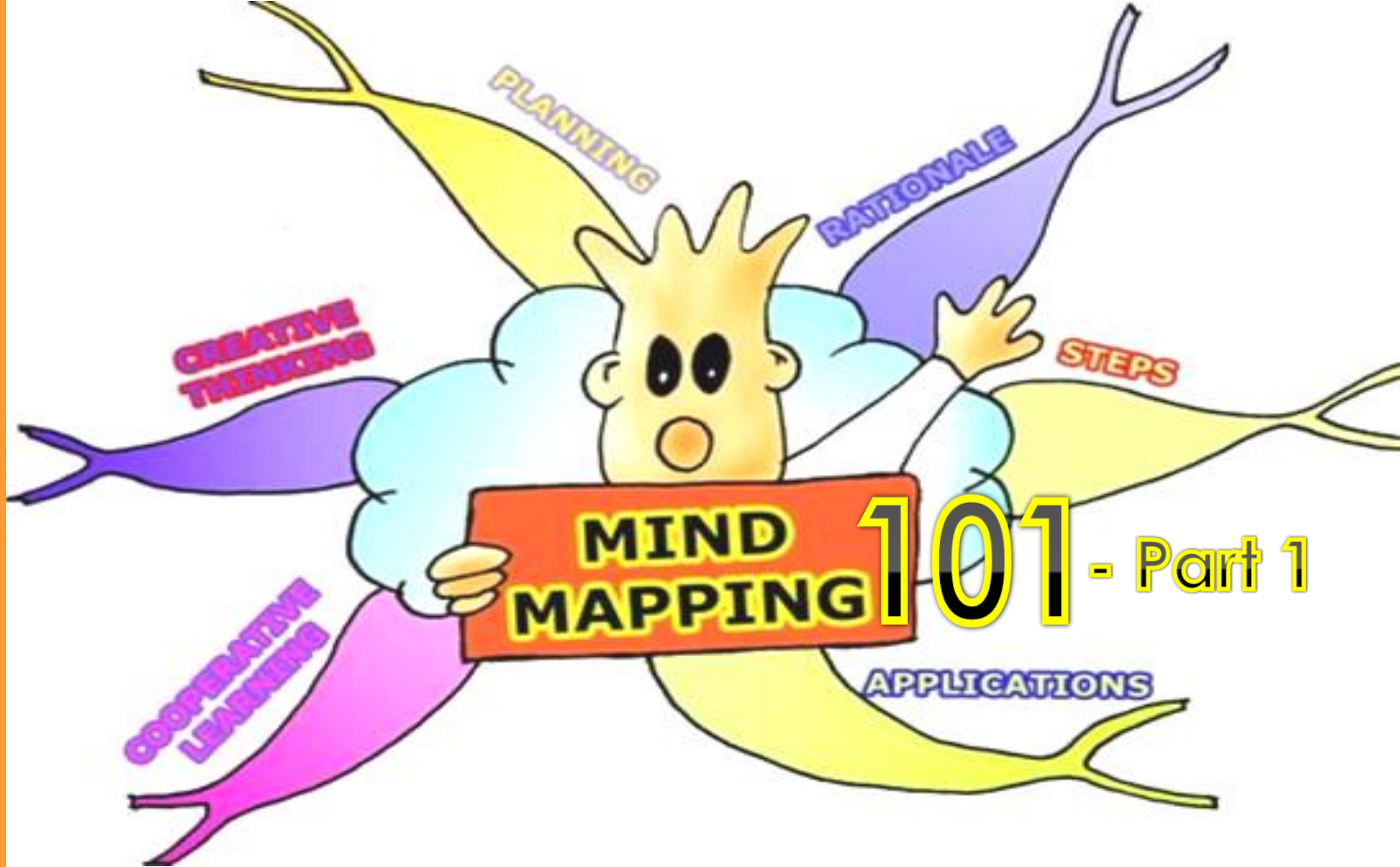
➤ Your Feedback or Question

Make sure to write **Feedback For < Article Name>** in your subject line.



In the school of Testing

for your better learning & sharing experience



By Darren McMillan

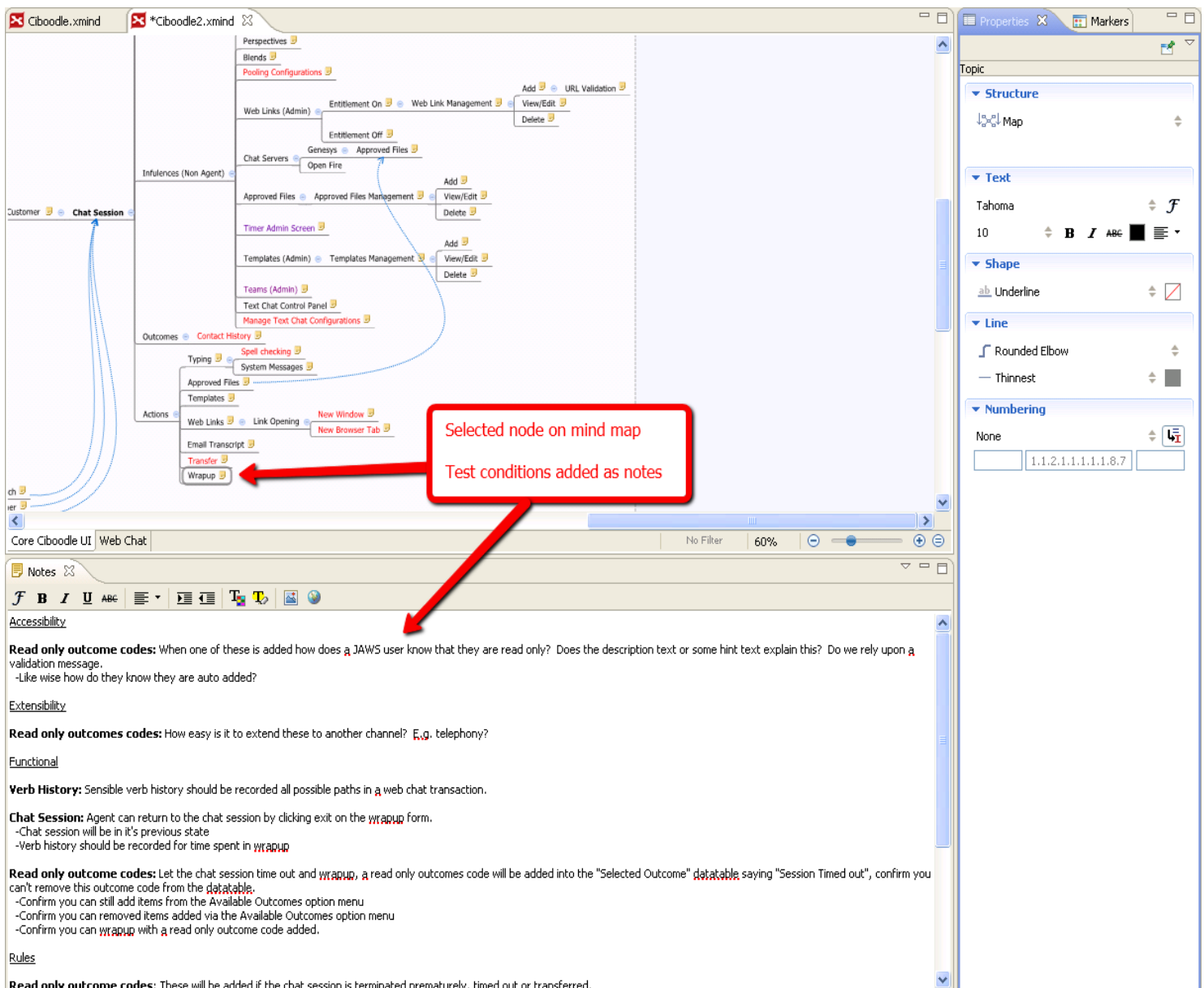
In this article we will look at the usage of mind maps for various testing activities such as test design, test planning, session reports, requirements analysis, self organization, and so on.

I was asked for some examples of how and where we use mind maps. Happy to help I responded with some example descriptions of how and where I use them. I'd like to follow up on those descriptions I gave with some real actual examples of mind maps and their usage; you can call this my **Mind Mapping 101**.

Test design

Mind maps can be fantastic tools to aid designing test cases for new or existing requirements. If done correctly you'll be able to produce higher coverage and better test conditions. My previous article, [Lean Test Case Design](#), demonstrates first hand how to produce rapid, lean, higher coverage and more efficient test cases using mind maps.

(Note : Click on the image to see it large)

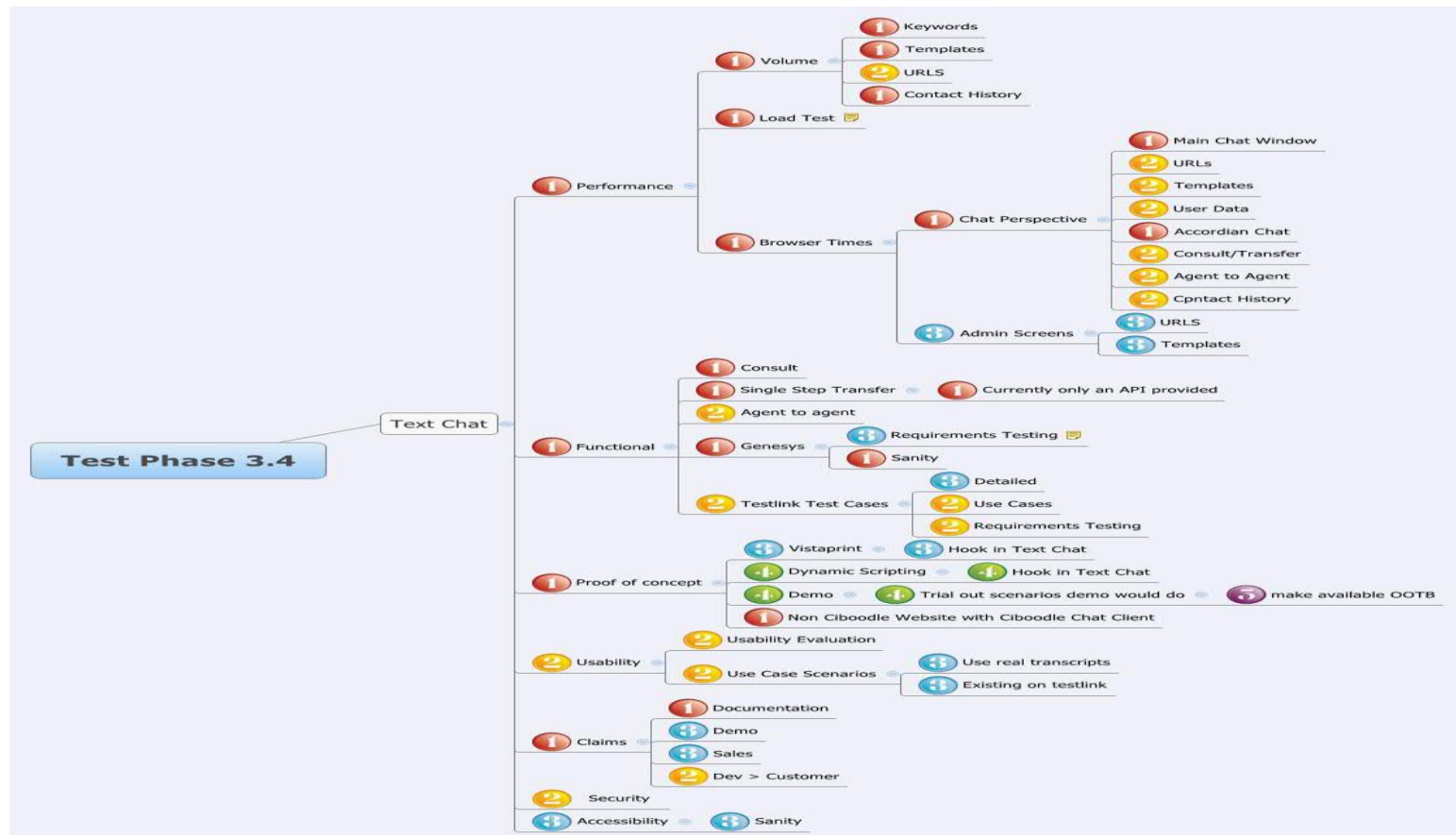


Test planning

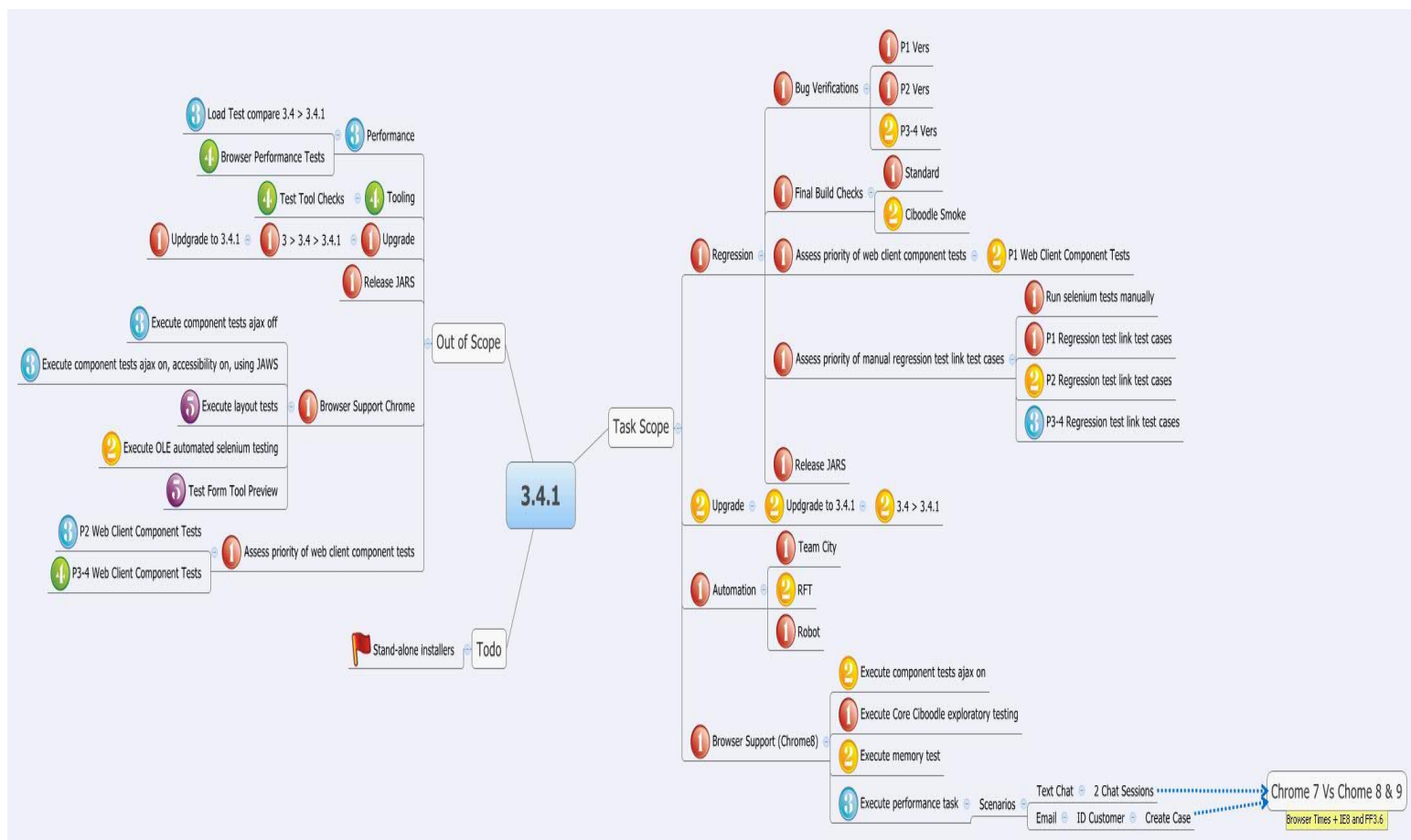
When planning scope for a new project, test phase or an extensive task I find mind mapping it gives me a platform to generate ideas more efficiently. From discussing the scope of the mind map with other key stakeholders the map will quickly evolve, with new tasks being added and unneeded ones being de-scoped. Finally once the scope has been agreed the map will either stay as-is, or be converted into a plan, or list of tasks on a task management system.

My post on [lean test phase planning](#) covers my approach to using mind maps when planning test phases.

(Note : Click on the image to see it large)



The mind map below demonstrates another test phase being planned with actual de-scoped tasks included for later reference.



Collaborative mind mapping

We all know the benefits of collaborating on plans, ideas, analysis and so on with others. Mind mapping is no different. In fact by discussing topics that require group thought with the aid of a whiteboard and some pens you can quickly draw up thoughts or ideas on a mind map. In fact you'll quickly find by mind mapping it, the process of developing new thoughts or ideas will come more easily.

As I don't photograph these, I can only provide one recent example provided by my boss **Michael Johnston**. This map displays our attempts to draw up our testing debt going from one release to the next.



(Note : Click on the image to see it large)

To be continued in Next issue...



Darren McMillan has been working in the testing field for over four years now. He has a genuine passion for testing all things and actively seeks to solve the problems others tend to accept. He strongly believes that opportunities are there to be taken and actively promotes self learning to others. When he is not testing or writing about his experiences, he enjoys nothing more than some quite family time.

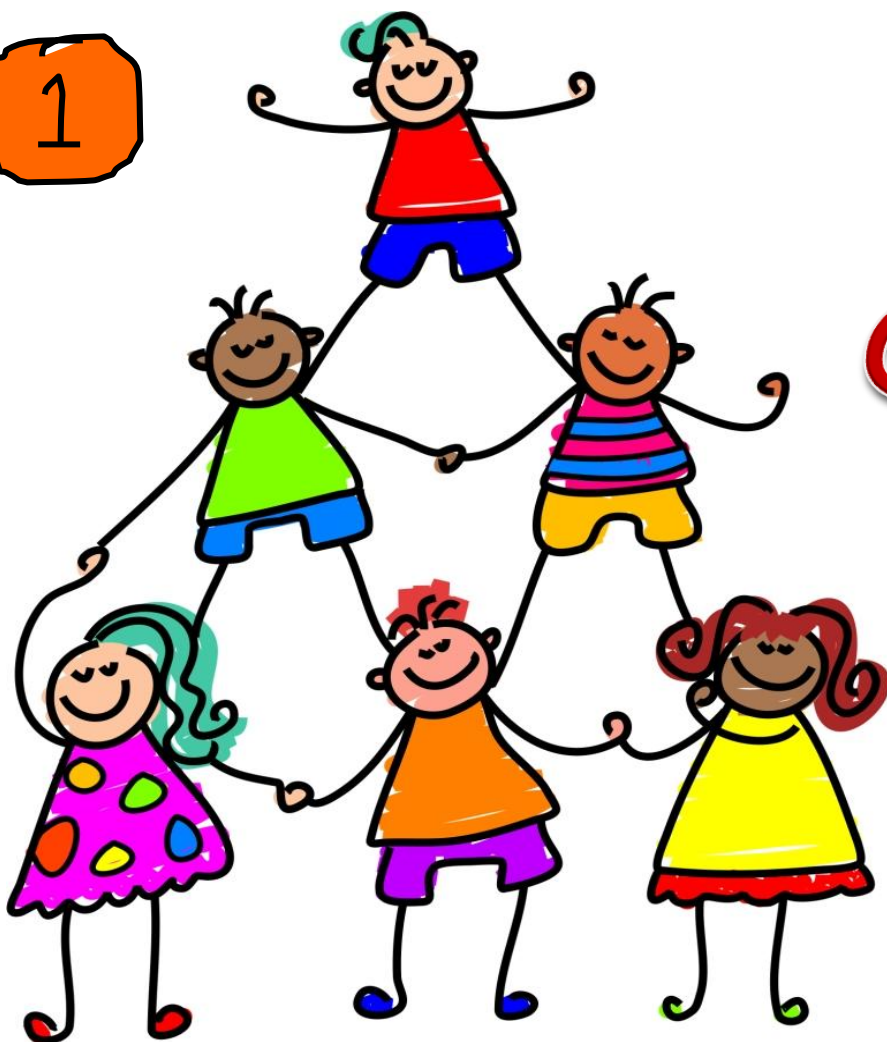
A proud father to a beautiful daughter he hopes that from leading by example he will encourage her to follow her own dreams. Contact Darren on Twitter @darren_mcmillan.



"Career Development and Learning Strategies for Testers" is a series of articles providing different approaches to develop testers' skills and knowledge from both a managerial and tester perspective.

By Bernice Niel Ruhland

1



Onboarding New Testers

The focus of this article is onboarding newly hired testers into your department. If you are not a hiring manager, but a tester who is changing companies or departments, you can identify information from this article to create an informal onboarding program.

It is important to integrate new testers to your department and company's social norms; policies and procedures; and communication protocol. The initial goal is for the employee to confirm that he made the right decision to join the company or transition to a new department. An onboarding program for an employee changing departments is also important to ensure he understand both departmental and job role differences. The second goal is to provide an initial training program to help him gain any skills and knowledge initially required for the job.

The onboarding strategy can be translated to a light-weight learning plan customized to the tester's training needs providing a foundation of expectations for the first few months. A learning plan coupled with conversations and hands-on assignments can help in both integration and socialization aspects.

Suggested Elements of a Learning Plan

There is not one approach that is correct for all companies and testers. The best way to approach the learning plan is to create a light-weight document with sections that are fairly standard and sections that are customized to the new tester. An advantage of incorporating standard sections is the manager does not need to recreate that information for every new tester. Below is suggested information that can be part of this plan.

Building a Social Network

Identify an employee to mentor the new tester for the first few weeks who will be available to answer questions and provide direction when appropriate. This is an important transition step until he builds an internal network of contacts plus it reduces some of the uneasiness in approaching other employees with questions. Try to remove the "I don't want to bother you" concerns. Also, it is often easier for a new employee to go to a peer than to his manager. Meet with the employee acting as the mentor so he understands his role and encourage him to seek out the new tester so it does not become a one-way communication path.



Identify key people he should meet such as developers, product managers, and business analysts. Schedule a 30-minute meeting with these employees to allow them to share their role and how they might work together. Be sure to identify co-workers the tester will be working with in the short-term.

Policies and Procedures

Gather any information that will be helpful such as policies and procedures. If possible, keep this documentation to what the tester will initially require such as the protocol to write and submit bug reports. Be sure to review this information with him to discuss the process and cultural norms such as discussing a bug with a developer before submitting to the bug tracking system. Clarify any interim

training steps such as submitting a bug report for review by another tester before discussing with a developer.

Professional Reading

Based upon the new tester's experience level, identify initial articles for him to read. It is important for testers to keep up with their professional reading to stay current with trends. For someone who is new to testing, identify articles such as "Testing Intelligence: Writing a good test case" by Joel Montvelisky published in Tea-Time with Testers, March 2011 issue.

For more experienced testers James Bach's article entitled "Investigating Bugs: A Testing Skills Study" published in Tea-Time with Testers, October 2011 issue might be appropriate.

Training Material and Initial Project

Identify any training material or resources that will help him learn the product under test. If the tester will be assigned to a specific project, be sure to provide any documentation. If the project has started, schedule meetings with team-members who can provide a project overview, review timelines, and provide a demo. If appropriate, have him shadow another tester on the project to get up to speed. Ensure that his role on the project and any initial expectations are clearly defined and communicated.

Identify Learning Sources and Technology

Provide a few free resources he can subscribe or follow such as Tea-Time with Testers and Michael Bolton's DevelopSense blog. In addition, identify areas within the company to obtain information such as the department's wiki page or Intranet page. Describe any tools used by the testing department such as screen-capture tools; location of the bug tracking system; and any other important tools. Identify employees who can work with the new tester to set up his testing environment and any other tools.

Define Expectations

Briefly define expectations for the first 60- to 90-days. If the company performs a probationary review, the timeframe should coincide. Defining expectations will help him know how to allocate his time and focus. For example, if an employee is new to testing, his focus may be on learning how to perform boundary, negative, and positive testing by working on a small project with daily oversight. A more experienced tester may have an extensive role on a larger, complicated project with little oversight.

Departmental Changes

When an employee change departments, it is important to address how his role in the company changes and how the testing department operates differently than the previous department. For example how they communicate with certain departments may change as to their role and information they provide. Requesting time off may follow a different process to ensure appropriate level of testing coverage. This information may be provided through both conversations and the learning plan for areas where written documentation is helpful.

The Testers's First Day

As the hiring manager, make time to meet with the new tester to discuss expectations for the first few days and cover initial social norms of the company and department. For example, the testers may schedule their lunch break to ensure there is sufficient testing coverage. Make sure the employee is introduced to the other testers and employees who are in the immediate area. Whether the learning plan is shared the first day can be dependent upon other scheduled activities and if it will not be information overload for the new employee.

Tips for the Manager

- Make the learning plan manageable; do not overwhelm the tester with too much information. Additional material can be provided at a later date.
- Gather information during the recruitment process to understand the tester's skill level plus ask him questions to determine any concerns about the position that can be bridged in the learning plan.
- The learning plan should be customized to the tester; however a portion of the plan can be static such as how to document bugs.
- The learning plan will not cover everything and new needs will arise, so periodically ask the tester what else he requires. Ask open-end questions to further understand his training progress and to determine if you need to change direction from the original plan.
- Allocate sufficient face-to-face time with new employees. It is fine to schedule meetings but also encourage informal conversations to ensure questions and problems are being addressed in a timely manner.
- Make changes to the learning plan template based upon what is learned from each tester to help onboard the next tester.
- Do not use the learning plan as a replacement for face-to-face conversations. If done correctly, the learning plan should foster questions and conversations. The manager has a responsibility to ensure the tester is properly integrated into the testing department, training needs are being met, and assisting the tester in building a social network.

Tips for the Tester

- **If a learning plan is provided:**
 - Review the plan and ask your manager any questions on expectations or other areas that might be unclear. Do not be afraid to ask questions as it is better to clarify expectations upfront.
 - Understand if there are any time constraints to complete the assignments and if certain areas are more important than other sections to initially complete.
 - Periodically review the plan for progress, to identify new questions, and determine any problems in completing the assignments which then needs to be discussed with the manager.

- Take ownership for your own professional development by going beyond the learning plan to identify additional learning opportunities.
- **If a learning plan is not provided:**
 - Review this article to identify questions that will bridge the gaps. Plus identify any additional information required based upon your particular job function.
 - Ask your manager for a 30-60-90 day expectations to reduce surprises at a later date; especially if the company performs a probationary review.
 - Take ownership for your own professional development by identifying additional questions and training requirements.
 - Create a learning plan, which can simply be written in a notebook or any preferred medium. Keep track of completed learning opportunities to show progress for both yourself and manager.

Conclusion

An onboarding program for testers is important for both newly hired testers to the company and for those transferring from another department. An onboarding program is designed to provide initial expectations and training for the first few months. The content of this program can be translated to a light-weight learning plan which fosters communication and provides initial assignments. Managers typically have the responsibility to develop a learning plan; however, a new employee can also take responsibility to work with his manager to create his own, informal plan. Regardless of the approach adopted, the employee should take ownership for his own professional development and not completely depend upon his manager. Whereas the manager should ensure that the new tester is properly integrated into the testing department, help foster building a social network, and ensure that initial training needs are met.

[Back To Index](#)

Bernice Niel Ruhland is a Software Testing Manager for a software development company with more than 20-years experience in testing strategies and execution; developing testing frameworks; performing data validation; and financial programming. To complete her Masters in Strategic Leadership, she conducted a research project on career development and onboarding strategies. She uses social media to connect with other testers to understand the testing approaches adopted by them to challenge her own testing skills and approaches.

The opinions of this article are her own and not reflective of the company she is employed with.

Bernice can be reached at:

LinkedIn:

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

Twitter: bruhland2000

G+ and Facebook: Bernice Niel Ruhland





are you one of those
#smart testers who
know d taste of #real
testing magazine...?



then you must be telling your friends about ..



Tea-time with Testers

Don't you ? 😊



Tea-time with Testers !

first choice of every #smart tester !



Testing...



with Anurag

Designing Test Cases for Mobile Applications: Things to Consider

Biography



Anurag Khode is a Passionate Mobile Application test engineer working for Mobile Apps Quality since more than 4 years.

He is the writer of the famous blog **Mobile Application Testing** and founder of dedicated mobile software testing community **Mobile QA Zone**.

His work in Mobile Application testing has been well appreciated by **Software testing professionals** and **UTI** (Unified Testing Initiative, nonprofit organization working for Quality Standards for Mobile Application with members as Nokia, Oracle, Orange, AT & T, LG Samsung, and Motorola). Having started with this column he is also a Core Team Member of **Tea-time with Testers**. Contact Anurag at anurag.khode@hotmail.com

Unlike web applications, Mobile applications have some peculiarities which need to be considered while creating the test cases for Mobile Apps.

Here is a simple formula which I use to follow while creating the Test Suite:-

Test Suite for Mobile Application=UI Test Cases + Functional Test Cases + External Factors Test Cases (Impact of app on native device functionality) +Performance Test Cases (If required) +App Submission/Third Party Certification Test Cases+ Usability Test Cases

This simple formula can help you to create a complete Test Suite which overall covers all the aspects of Testing of any mobile App.

Based on my experience, I have derived some points which may help you to create effective test cases for Mobile Applications.

Few things to consider while writing Test Cases for Mobile Apps:-

1. Identify UI Test Cases: -

User interface is one of the most important aspects for any application and so as it is true for mobile applications also. While writing test cases for Mobile Apps, it is important to identify test cases for User Interface also. The test cases can be in the form of verifications points making sure that application is developed as per Mock up provided by client(OR Design Team).Info/Error message UI, Screen Color/Theme, Application Logo, readability of content, Menu Style, UI – Screen repainting etc. should be considered while testing the User Interface of Mobile Apps. For some platforms/OS like iOS(Iphone),it is advisable to go through Standard guidelines ([Human Interface Guidelines](#)) to understand the platform specific UI requirements and verification points should be included for the same in your UI Test Cases.

2. Consider Low level Test Cases also: -

I have always seen people getting confused in involving some low level test cases in their test suite. Smooth Navigations among different media or navigation of focus over the controls, Pagination, sorting, Resetting the focus on immediate object after deletion of any object, Scroll Bar positioning, verification for input box limit, Menu/Submenu, Truncation ...well there are many small but important things to consider in mobile apps. It is always better to have a small checklist for this too in your test suite. Either you can have a separate checklist or sometimes you can involve such test cases in Expected Result also. Eg. in following case:-

Test Case Objective: - To make sure that contact is deleted from the contact list

Action:-

- Move the focus over the contact
- Press LSK and select menu option "Delete"
- Select "Yes" on confirmation message

Expected Result:-

- The contact should be deleted from the list.
- **Focus should be retained on immediate contact**

In the above example, you can see that it is made sure that focus is retained on the immediate contact in expected result itself. The reason why these low level cases are important is that, many times these cases are ignored and finally it overall impacts the usability of application. **For example** you are testing

an application having collection of pictures (Image Thumbnails). Now user can move to next page by some option to see some more images. Now if user is on 6th or 7th page, deleting any image reloads the page and user is navigated to first page with focus on first page. Now if user has to delete or view some more images from 7th page onwards, user again needs to move one by one from one page to other till he gets 7th page. Here surely user may get irritated and he may not be interested to use this (delete) feature of your application so frequently. So though the major test case of deleting the thumbnail is passed, but it is also important to take care of some low level cases (For example retaining the focus on immediate thumbnail here) which may hamper the overall usability of application.

3. Consider Network Related Test Cases:-

Tester must include effective test cases related to Network connectivity while designing test cases for mobile apps e.g.

- To verify the behavior of the application when there is no network.
- To verify the behavior of the application when network is weak.
- To verify the behavior of the application (e.g. downloading of content) when network is resumed back.
- To verify the application behavior on different networks like EDGE, GPRS, Wi-Fi, 3G.

While designing the test cases for Network related cases, please don't forget to consider both Foreground and Background Data Calls in the application. For example:-

- To verify the case when downloading of video is going on and Network is lost.
- To verify the case when downloading of video is interrupted due to No/Weak Network and Network is resumed back.
- To verify the case when Uploading of content is going on in Background and Network is lost.
- To verify that interrupted uploading process of content is resumed back when Network is re-established.
- To verify the case when user is signed in to any service (say Mobile yahoo Messenger), accessing any static screen (Say About Screen) and Network is lost.

In short there are many scenarios which need to be considered as a part of Network test cases of mobile apps. As there are many screens and many dynamic activities in the mobile application, it is preferred to create a verification table for different screens and dynamic activities in the application. Please refer table used in Point #7 here in this article.

4. Consider Performance Factors :-

Unlike web application, performance has slightly different aspect in mobile application. While designing the test cases for mobile apps, it is always preferable to create a separate sheet to document the response time for any activity which may take time in the app. This is to monitor the performance for any activity/action in the application. This may include noting the time required to load 100 thumbnails (Say) in the application or time required to perform any action.

While collecting the data, make sure that you are also documenting the device details on which test is conducted for e.g.

- Device Name
- Firmware/OS Version
- Memory Status
- Network Type (Edge, GPRS, 3G etc.)

Sr. No.	Activity	Time Taken(Sec)	Avg. Time (Sec)	Bench mark Time	Remark
1	Launch Application	5	4.16	<3 Sec	
		3			
		4.5			
2	Delete Thumbnails (1)				
3	Delete Thumbnails (100)				

5. Include Third Party Certification Test Cases :-

I am not sure that how many of you are aware of this but for some mobile platforms, for example BREW , you need to certify your application against True Brew Testing Criteria. Your application needs to be certified by NSTL against True Brew Test Cases(Set of test cases which needs to be passed in order to launch the application).Hence, It is always advisable to have such test cases as a part of your test suit. Similarly if you are launching your application in any Application store, they may have their own set of criteria which need to be satisfied. As a smart Mobile App Tester, you should include these test cases in your test suite. You may observe that most of the test cases are already covered in your Functional Test Cases, but still you will find that there are many test cases which were not covered yet in your test suit.

6. Consider Screen Orientation:-

Unlike feature phones, now most of the Smartphone supports change in screen Orientation. Verifying UI Repainting when there is change in orientation from Portrait to Landscape and Landscape to Portrait is important. Tester should make sure that he has verified each and every screen for screen orientation. Change in orientation mostly affects the screen UI of the application.

However in some cases it may hamper functionality also as due to distortion in UI, user may not be able to use any feature in the application. Here is an example about how you can maintain a sheet in your test suite to make sure that you have verified Screen Orientation on each and every screen in Application

Sr. No.	Screen Name/Activity	Verified (Yes/No)?	Result	Comments
1	Splash Screen	Yes	Pass	
2	Login Screen	Yes	Fail	
3.	Media Player Screen	No	NA	
4	Download Progress Bar	Yes	Pass	
6.	Info/Error Message	No	NA	
7.	Popup Menu/Submenu	Yes	No	

To be continued in Nest issue.....

Test Case writing in Mobile Apps has different Horizons and considerations. In My next article, I will focus on some more aspects of writing test cases for mobile apps. Till then stay tuned ☺ .

[Back To Index](#)



testing intelligence

- *its all about becoming an intelligent tester*



an exclusive series by **Joel Montvelisky**

Switching to Agile Testing, not as simple as changing your t-shirt

In addition to all my other tasks in PractiTest, I am a tester working and consulting on a number of Agile projects. In total I've been in testing for over 15 years and doing agile testing for close to 5 years.

Here's a fact some people tend to overlook, just like any other profession, to become an expert on a specific field of testing you need to work hard and develop the skills required for that specific type of testing. And so I know some very impressive testers that are experts in the field of Load Testing, I also have a number of colleagues who I define as Automation Experts (having the strange ability of been good testers and good developers at the same time), I also know some pretty impressive Exploratory Testers, as well as a number of Security Testers, Usability Testers, etc. Each one of these sub-especializations comes with their own challenges, added value, tasks, etc.



Agile Testing is also a sub-especialization of testing, that comes with some pretty singular challenges, and requires specific skills that a tester needs to develop in order to succeed in his job.

Without underestimating non-agile testing, I find that testing on agile projects can be a lot more challenging and demanding than been part of, let's say, a Waterfall testing team.

In fact so demanding, that lately I have seen a number of testers that were very valuable and effective working on non-agile projects walking-out or been laied-off from teams working on agile organizations, because they just didn't have the skills required to make the transition into Agile Testing.

What's so demanding about Agile Testing?

There are a number of things that make agile testing challenging. I think the main ones (or at least the ones I see more people stumbling and failing) are: becoming an organic part of a development team, finding yourself as the "lone tester" in charge, the "crazy" pace of Agile development, and the need to work with a slim process & documentation and the risks involved with this.

1. Becoming an organic part of a development team is demanding because of the intrinsic rivalry that usually surrounds the relationship between developers and testers in non-agile teams. Even if we all have the same objective and target, each of us approaches their jobs in a different (and sometimes opposing) angles.

Agile testers and developers need to learn how to work together in one team. When you are part of the same organic team it's not enough to have the same goals and objectives on the "MACRO level", and we need to find the way coordinate our tasks and jobs in the "MICRO level" as well.

This can be frustrating for both developers and testers, and somehow (maybe because of the cultural baggage that we are still carrying from pre-agile times) testers always feel they will need to concede to developers even when they don't agree with it.

2. Finding yourself as the "lone tester" in charge is challenging because all of a sudden you need to work alone, calling the shots and taking the risks in front of your programming peers.

It may sound strange, but there is a big advantage that comes from working within the "protection and support" of a group of testers that, when push comes to shove, will stand next to you and defend your "testing-based" approach. For someone who has not been in a leadership position (as a team lead or manager) this can be very intimidating.

This issue is specially acute if the Agile Team Lead and the Scrum Master also lack the experience and understanding to know that they need to provide backing and support to their testers...

3. The crazy pace of agile development is hard to all the team, but it is specially challenging for testers that continue doing a big part of their testing tasks towards the end of the process (even if we work on short sprints).

This issue becomes even more acute because when you work on an agile team, towards the end of the sprint you are supposed not only to run your most critical tests but also to "employ" your fellow developers in order to get some help in the testing tasks, and we all know that managing developers to do testing tasks is not easy (not even in the remote possibility that they *really want* to help...)

4. Working with a slim process & documentation and the risks involved is the fourth challenge for testers transitioning into Agile.

Whether we like to accept it or not, many of us use process as a way of controlling our projects. When the project is under control we have a better chance of discovering and managing most of its risks.

The same goes for documentation, when people need to document their work (and when they do it correctly!) they tend to find more issues and discover more risks earlier in the process.

Because agile projects work with slimmer processes and very limited documentation, then the responsibility of the tester to find and manage the risks of the project becomes more important but at the same time more challenging, since he needs to find alternative mechanisms to detect these risks and control them with the rest of the team.

What can you do about it? Should you stay away from Agile Testing?

Definitely NOT!

Many people, including myself, find agile testing to be extremely fun and gratifying. It provides us with challenges that expand over the methodological, process and technical realms; and allow us to influence the development process even more than working in more traditional “V-Model” processes.

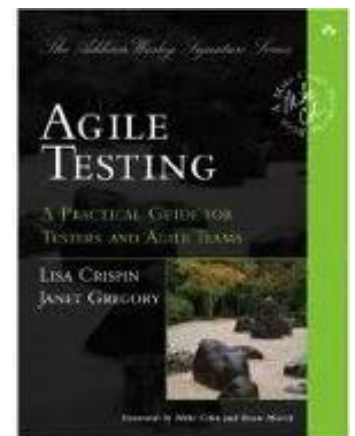
The best thing to do if you want to transition into an Agile team is to make sure you understand what you are getting into. Talk to agile testers and if possible go and see how they work. Make sure you can see yourself working and enjoying the “organized chaos” that comes with agile projects.

Once you decide Agile Testing is what you want to do, work on developing your agile testing skill. How? You can “jump into the water” and start working on agile projects, but take it one step at a time.

Another thing you can do is study. Start by reading some good books, like **Agile Testing** (by Lisa Crispin and Janet Gregory) where they explain how agile testing projects work, and how a tester can succeed on them.

I specially like their 10 principles for agile testers:

- Provide continuous feedback.
- Deliver value to the customer.
- Enable face to face communication.
- Have courage.
- Keep it simple.
- Practice continuous improvement.
- Respond to change.
- Self-organize.
- Focus on people.
- Enjoy 😊



You can also find countless blogs and articles on the subject by simply googling “Agile Testing”.

What do you say?

Do you have something to share from your transition into agile? Please come and share it!

Is your experience of transitioning into agile different?

What tips do you have?



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

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

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

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

Joel publishes a blog under - <http://qablog.practitest.com> and regularly tweets as [joelmonte](#)

[Back To Index](#)

Teach-Testing

An Ambitious Campaign by Tea-time with Testers



[Click here to Cast Your Vote](#)



by





Call for Articles !

Have you got something to say?

yes, we are listening you...!!!

"Tea-time with Testers" firmly believes that one of the best ways to improve upon software testing is to listen to the lessons learned by others and their experiences too.

So, if you have an interesting story that you'd like to share with the world, contact us at teatimewithtesters@gmail.com.

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

Click [HERE](#) to read our Article Submission FAQs !

T ' Talks



T. Ashok exclusively on software testing

Form and Structure MATTERS! A roller coaster ride experience

Riding a roller coaster is great fun. Irrespective of age the ride is filled with thrills. The rush you get from roller coaster is indeed out of the world. The slow climb and then a sharp fall with gut wrenching twists and turns is an exhilarating experience.

We look at this engineering marvel with AWE and RESPECT. How is it possible that this steel lattice manages to give so many thrills and still hold a safety record, better than a toaster?

The answer is simple—it's the form and structure of the roller coaster. The physics behind the form and structure are key to the forces that we experience which contributes to the thrills we enjoy.

"Form and Structure" is the integration of design and engineering. It is ubiquitous, from nature's leaves and honeycombs, to man-made structures. Form is about the external shape, the way it is presented, while structure is about the way the elements are arranged/composed. Both are critical to any design activity and this applies to test design too.

Typically our belief is that effective testing is the result of good test cases. The "goodness" of test cases is normally associated with the test case contents. **Are good test cases the result of one's experience or the strength of the techniques only?**

The architecture consisting of the external form and the internal structure play a vital role in ensuring that the test cases are adequate yet optimal and can be generated and automated rapidly. The form gives the shape to test cases allowing us to see a variety of dimensions related to adequacy, reviewability, automate-ability, and product health. Structure is what allows assembling the various elements of a test case to create the shape that allow seeing the various dimensions.

Hypothesis Based Testing (HBT), the personal scientific test methodology pays significant attention to the form and structure of test cases. The test cases in HBT has a nine-dimensional form (or shape), with each dimension focusing on a specific attribute of goodness of the test case. The nine dimensions are:

1. Cleanliness level - What level of quality or cleanliness are these (test cases) focused on?
2. The entity they are validating - Is it at an elemental component, technical feature or business flow?
3. Potential defect type it is expected to uncover - What type (or class of defect) is it expected to uncover? And the types of tests needed i.e. clear segregation of test cases into test types (for an entity for a given cleanliness level).
4. Its focus (conformance or robustness) - Does it validate the correct use or recoverability in the case of abuse?
5. Its priority or importance - The perceived importance of the test case to the overall customer experience
6. The stage when it needs to be executed - At what stage of development is it expected to be executed? Is it for validation of the entity or ensure build correctness or periodic regression to ensure that health is not compromised?
7. The intended execution frequency - How frequently do we expect these to execute? Daily, weekly, once a build, etc
8. The optimal sequencing or threading of test cases - Which is the next test case to execute when the current one fails? This is especially useful for test automation to ensure that we do not "baby sit" the scripts and ensure unattended long runs.
9. The mode of execution - How do we intend to execute - manually or automate it?

The first FOUR dimensions focus on efficacy aspects of test cases while the remaining FIVE dimensions largely focus on efficiency of execution of test cases. The segregation of test cases in the first two dimensions (of cleanliness level and entity) also allows rapid automation as an entity under test to be validated and consists of short scenarios that are staged by cleanliness levels making their conversion into scripts easier and simple to maintain.

Coming to the structure, in HBT there is a clear notion of behavioral scenarios for each element under test, with each behavior assessed by a set of stimuli i.e. test cases. In addition to the structural aspect of requirements traceability, there is an interesting notion of FAULT Traceability that associates each scenario with the potential defect that they can uncover. At the lowest level, structure focuses on the simplicity or the terseness of how they can be written, to ensure rapid test case design yet being very effective.

Test cases are not merely a list of execution steps; the key is optimal combination of inputs that form the stimuli to evaluate the behavior of an entity under test. Tracing the stimuli to potential types of defects it can uncover, makes the test cases purposeful and effective. Finally, detailing the preconditions and execution steps aids in converting these into automated scripts.

The contents of the structure can be documented for posterity in a form that is terse or detailed; this is probably governed by the organizational process.

Form and structure are integral to the goodness and should not be confused with documentation detail. After all a beautiful rose can be described in a single sentence or in an elaborate poetic form.

In summary, the form and structure enables

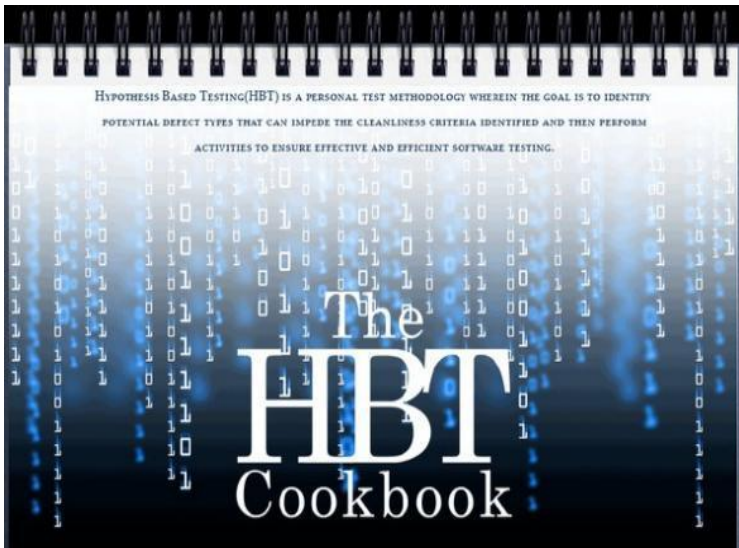
- Test cases to be sharply goal focused i.e. what types of defects to uncover
- Clear assessment of effectiveness of test cases
- One to select appropriate test cases to optimize execution
- One to objectively assess the "system health"
- Development of shorter scripts aiding rapid automation with low maintenance

So when you design test cases, do not just focus on the contents, see if there is a form and structure that allows us to see the various dimensions of goodness. Ultimately a good form and structure brings about the aesthetic value, help to see and enjoy beauty in what we do, and get a satisfying experience rather than just test!

On a lighter vein, the next time you ride a roller coaster, do not think about testing - ENJOY the ride!

Have a nice day.

[Back To Index](#)



If you are keen to know more about HBT, click on the adjacent image.



Experience the roller coaster ride.

Watch this short video below -

Biography



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.





OUR PARTNERS

Quality Testing



Quality Testing is a leading social network and resource center for Software Testing Community in the world, since April 2008. QT provides a simple web platform which addresses all the necessities of today's Software Quality beginners, professionals, experts and a diversified portal powered by Forums, Blogs, Groups, Job Search, Videos, Events, News, and Photos.

Quality Testing also provides daily Polls and sample tests for certification exams, to make tester to think, practice and get appropriate aid.



Mobile QA Zone

Mobile QA Zone is a first professional Network exclusively for Mobile and Tablets apps testing.

Looking at the scope and future of mobile apps, Mobiles, Smartphones and even Tablets, Mobile QA Zone has been emerging as a Next generation software testing community for all QA Professionals. The community focuses on testing of mobile apps on Android, iPhone, RIM (Blackberry), BREW, Symbian and other mobile platforms.

On Mobile QA Zone you can share your knowledge via blog posts, Forums, Groups, Videos, Notes and so on.







Service Test

PART 2

By Karthik Subramanian

Testing Web Services using HP Service Test

Introduction to Web Service

Web Services is the standard for integrating application over the Web. It is based on open technologies like XML, SOAP and WSDL.

Unlike traditional Desktop or Web based applications, Web Services do NOT provide a GUI interface. XML is used to describe the Web Service, SOAP is the Protocol and WSDL is used to describe the Web Service.

As you can see it can be quite complex to deal with these technologies and effectively test a Web Service. In this article, we will simplify this process by using HP Service Test to automate this testing process

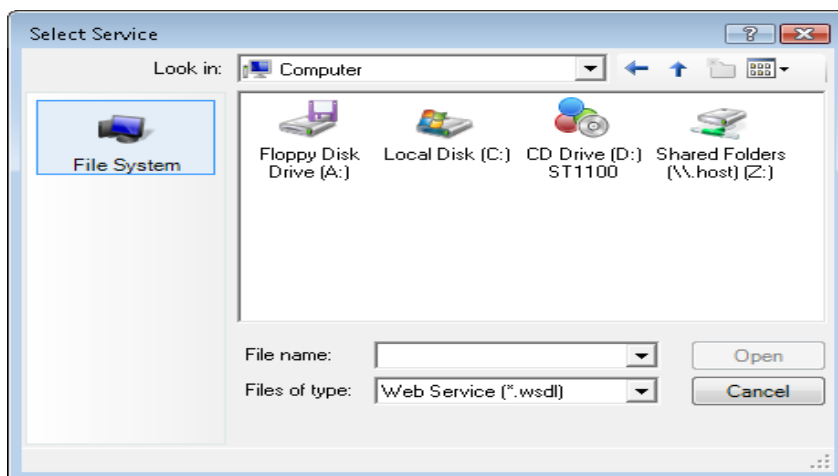
Basic Steps to work with web service

Click  Import WSDL in the toolbar to import a WSDL Web Service

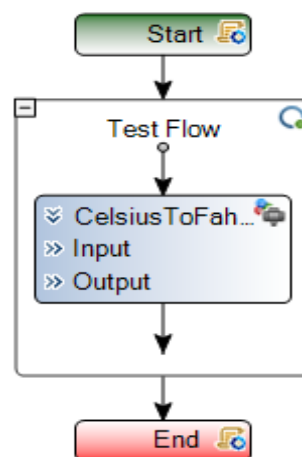
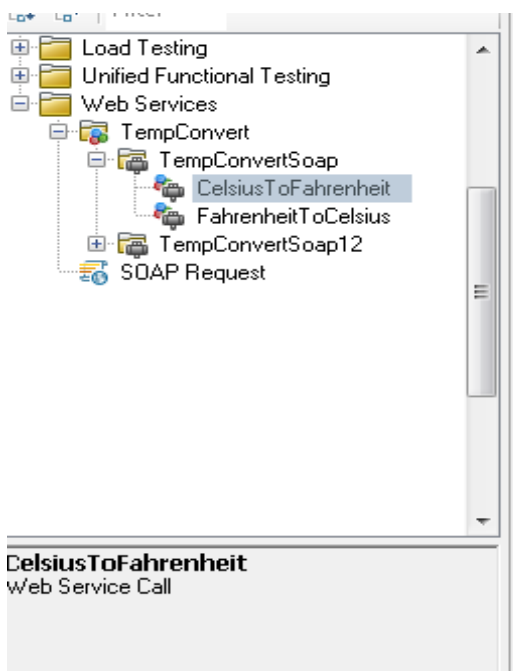
You can import WSDL Web Service using the URL / UDDI



You can also choose WSDL Web Service from the File System or QC/ALM



Toolbox palette will be loaded with the imported WSDL. In the toolbox palette you can drag the required activity into the canvas palette. In this example, we drag Web Services 'CelsiusToFahrenheit' to the canvas.



Assigning input data and setting checkpoint in the property sheet. The input data will be used in converting temperature and checkpoint data is to validate the conversion.

Input Value for Test (0)

Expected check point Value from Web Service (32)

The screenshot shows two windows from a test runner. The top window is the 'Property Sheet' for a test case named 'CelsiusToFahrenheit'. It has a tree view on the left with nodes: Envelope, Header, Any [...], Body, CelsiusToFahrenheit, and Celsius. The 'Value' column on the right shows the value for the 'Celsius' node is 0. The bottom window is the 'Checkpoints' tab. It has a table with columns: Checkpoints, a checkbox, and Expected Value. The table has one row for 'CelsiusToFahrenheitResponse' with a checked checkbox and an expected value of '32'.

Input	Value
Envelope	
Header	
Any [...]	
Body	
CelsiusToFahrenheit	
Celsius	0

Checkpoints		Expected Value
Envelope	<input type="checkbox"/>	=
Header	<input type="checkbox"/>	=
Any [...]	<input type="checkbox"/>	=
Body	<input type="checkbox"/>	=
CelsiusToFahrenheitResponse	<input checked="" type="checkbox"/>	Contains 32

Run and check the output for response and checkpoint validations in the output tab

Output

Run

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <CelsiusToFahrenheitResponse xmlns="http://tempuri.org/">
      <CelsiusToFahrenheitResult>32</CelsiusToFahrenheitResult>
    </CelsiusToFahrenheitResponse>
  </soap:Body>
</soap:Envelope>

Checkpoint0: "32" Contains "32" passed
Checkpoint1: "System.Xml.XmlElement" EqualToWithIgnoreNS "32" failed
Step 'CelsiusToFahrenheit' ended successfully
Step 'Iteration 1' ended successfully
Step 'Test Flow' ended successfully
Step 'End' : Step 'End' Started
Step 'End' : Parent Step: 'test'
Step 'End' ended successfully
END

```

The screenshot shows the 'Output' window of a test runner. It displays the XML response from the web service, which is a SOAP envelope containing a body with a CelsiusToFahrenheitResponse element. The response contains a CelsiusToFahrenheitResult element with the value 32. Below the XML, there are two checkpoints: 'Checkpoint0: "32" Contains "32" passed' and 'Checkpoint1: "System.Xml.XmlElement" EqualToWithIgnoreNS "32" failed'. The test flow ends successfully.

Checking Test Result for test validation

The screenshot shows the HP Run Results Viewer interface. On the left, a tree view displays the test structure: Test WSDLW3Schools Summary, Flow Diagram, Start, Test Flow, Iteration 1, CelsiusToFahrenheit, and Checkpoints (highlighted). The main pane shows 'Step Name: Checkpoints' with a 'Step Passed' status. Below this is a table with columns: Object, Details, Result, and Time. The table contains one row: Object: Checkpoints, Details: (empty), Result: Passed, Time: 10/3/2010 - 19:26:39. At the bottom, the 'Captured Data' section shows a table with columns: Name, Result, Property, Actual Result, Evaluation Style, Expected Values, and Details. The table contains one row: Name: 'Checkpoint0', Result: (green checkmark), Property: '\\CelsiusToFahrenheitResponse [1]\\CelsiusToFahrenheitResult [1]', Actual Result: '32', Evaluation Style: Contains, Expected Values: '32', Details: ''.

Object	Details	Result	Time
Checkpoints		Passed	10/3/2010 - 19:26:39

Name	Result	Property	Actual Result	Evaluation Style	Expected Values	Details
"Checkpoint0"	✓	"Envelope[1]\\Body[1] \\CelsiusToFahrenheitResponse [1]\\CelsiusToFahrenheitResult [1]"	"32"	Contains	"32"	""

[Back To Index](#)

Do you think that even your own tool should be part of this unique section?*

Feel free to write us. Let the world know what your Tool can do !

To know more write to us at teatimewithtesters@gmail.com

* Conditions Apply



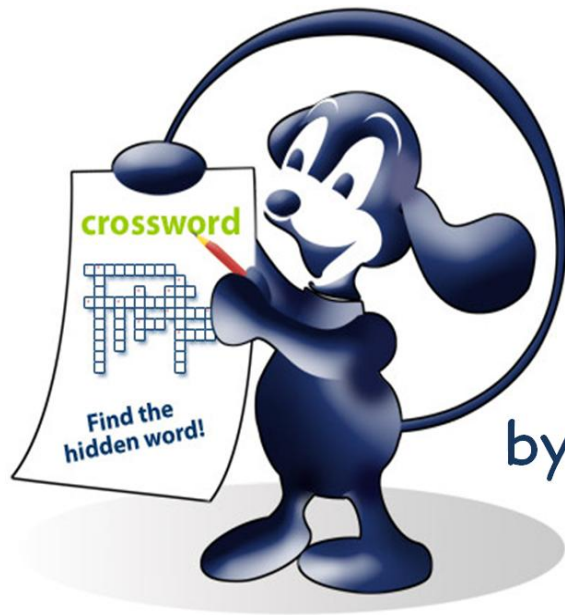
Karthik Subramanian is the Senior Technical Solutions Consultant at Hewlett Packard (HP). Prior to that he was with Mercury Interactive; serving as the internal Product Expert on Automation tools.

He holds numerous industry certifications including: Mercury QTP CPC, WR CPS, ISTQB, Sun SCJP and Microsoft MCP. He has also presented at numerous conferences including: HP Meet the Expert Sessions, HP Software Universe and Mercury Software World.

He also holds a masters degree from University of California, Davis and is a graduate of the Indian Institute of Technology (I.I.T).

Testing PUZZLES

by Sebi



Claim your **Smart Tester of The Month** Award. Send us an answer for the Puzzle and Crossword below b4 15th Dec 2011 & grab your Title.

Send -> teatimewithtesters@gmail.com with Subject: Testing Puzzle

Gear up guys.....

It's Time To Tease your Testing Bone

Puzzle “Play Around It Again😊”

Find the longest string that is used as a sub-domain for a website.

Example: "news" in "news.google.com". There should be no redirected sub-domains.



Biography



Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years. He is currently hosting European Weekend Testing.

He considers himself a context-driven follower and he is a fan of exploratory testing.

He tweets as @testalways.

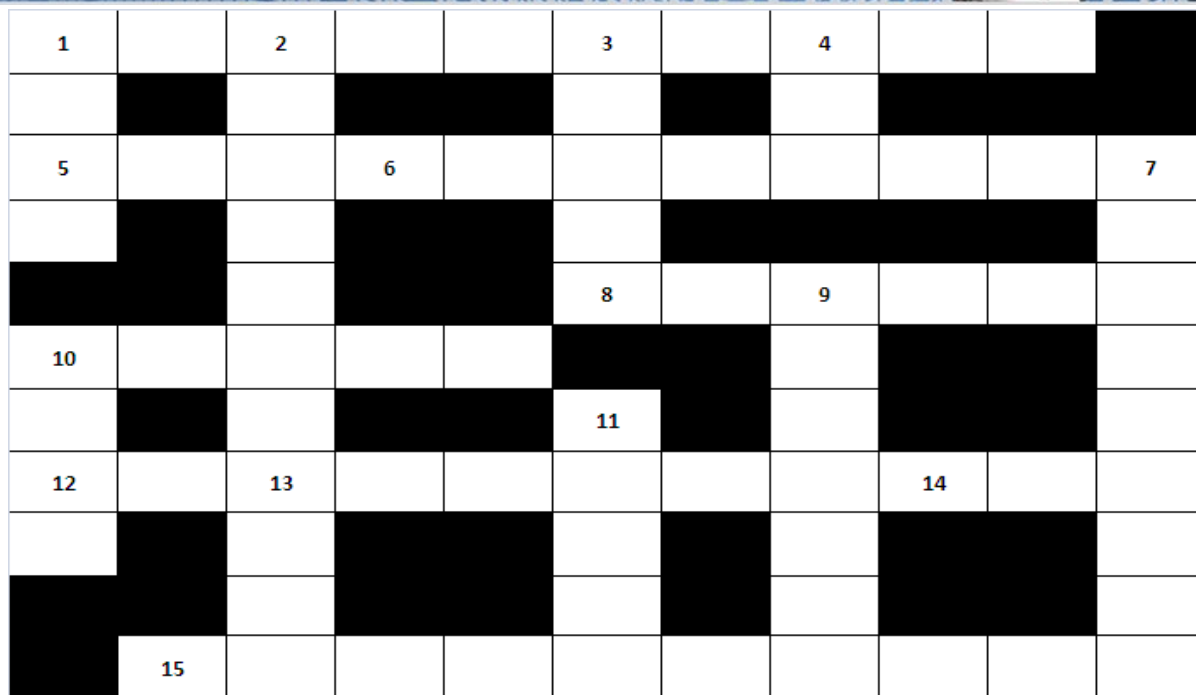
You can find some interactive testing puzzles on his website www.testalways.com

[Back To Index](#)





TESTING CROSSWORD



Horizontal:

1. Which company CEO Jeff Lusenhop has been awarded the honor of CEO of the year in the Small For-Profit category in the first Central Ohio CEO Survey, conducted by Columbus C.E.O. magazine and Capital University (6)
5. It is a tool/editor which can is used for designing purposes, mainly for object oriented concepts, like you can draw class diagrams, sequence diagrams etc, the basic architecture of a project, second word (4)
6. A test approach in which the test suite comprises all combinations of input values and preconditions, first eight words (8)
8. It is a graphical Eclipse plug-in for writing Selenium and Watir tests, first five words (5)
10. It is a free Open Source tool for automated testing of web applications in a very effective way, its first word (5)
12. A process of finding and reducing the number of bugs, or defects, in a computer program thus making it behave as expected is called _____? (9)
14. CEO of Testcover.com, in short form (2)
15. A type of review that relies on visual examination of documents to detect defects, is called _____? (10)

Vertical:

2. He (his first name) is a Winner for Testing Crossword and awarded Smart Tester of the Month of August (7)
3. Testing the product with no aim, no guidance and no goal (5)
4. A program interruption that occurs when a page that is marked 'not in real memory' is referred to by an active page, its called _____, in short form (2)
7. It is not unique to performance testing but a term used to describe repeating a test execution that resulted in an application problem. It is called _____?, first word (9)
9. The percentage of branches that have been exercised by a test suite. It is called _____?, first name (6)
11. The first word in Agile Testing (5)
13. It is a Ajax test runner for php, the first word (6)

Answers for Last Month's Crossword:

L	I	S	A		A		Q	A	
A			D	E	B	U	G		E
C	M	M	H		U		I	T	P
C			O	A	K		T		D
E	G	S	C		Y		A	N	N
P		Y					K		R
T	E	S	T		F	R	A	N	K
A		T			J				N
N		I			I		M		C
C	O	N	F	O	R	M	A	N	C
E					A		C		

Answers for Testing Puzzle of last month:

Correct Website : <http://code.google.com/webtoolkit/doc/latest/DevGuide.html>

Incorrect Website : <http://code.google.com/webtoolkit/doc/latest//////////DevGuide.html>



We appreciate that you

"LIKE" US!



Join us on Facebook.

You are just a CLICK AWAY



Every Tester

who reads Tea-time with Testers,

**Recommends it to friends and
colleagues .**

What About You ?

Our Testimonials

Great magazine!

Your magazine is a great read with lot of interesting things packed in it!

Your team is doing good job in getting this out neatly every month!

I forward your magazine to all those who can get benefit out of your articles.

- K.N. Keshava Murthy

Tea Time with Testers is a very creative name. I am very curious to how your magazine is.

- Deepthi

Good concepts, articles and ideas. Kudos !

- Rajesh Sutar

Genuinely a rewarding effort !

It's genuinely a rewarding effort. I have recently started to follow the news and activities through your publications.

The contents included are qualitative and covers the multiple dimensions of Software testing domain. Wish you great work ahead!

- Rupesh Dev.

This is my favorite testing magazine!

- Amy B

Great ! Tea-time with Testers is very different and unique testing magazine that I have seen. Hard to believe that you are doing so much for free. Thanks a bunch for your efforts.


- Sara Kingston

I came to know about your magazine when my lead shared it with me. Really liked it and I have become fan of it .

- Mudhumita Kelkar

You ask...

We'll help...!



If you have any questions related to the field of Software Testing, do let us know. We shall try our best to come up with the resolutions.

- Editor

Feel free to write us your expectations.
Help us to help you better.

We are just a mail away: teatimewithtesters@gmail.com

in ne>xt issue

articles by -

Jerry Weinberg

T Ashok

Joel Montvelisky

Darren McMillan

Anurag Khode

Karen Johnson

J DeMeyer

Bernice Ruhland

Mike Talks

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

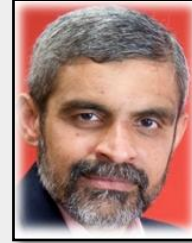
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover Page Image- Etsy

Core Team:

Kavitha Deepak (Bristol, United Kingdom)

Debjani Roy (Didcot, United Kingdom)

Anurag Khode (Nagpur, India)



Anurag



Kavitha



Debjani

Mascot Design & Online Collaboration:

Juhi Verma (Mumbai, India)

Romil Gupta (Pune, India)



Juhi



Romil

Tech -Team:

Subhodip Biswas (Mumbai, India)

Chris Philip (Mumbai, India)

Gautam Das (Mumbai, India)



Subhodip



Chris



Gautam

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

To get **FREE** copy ,
Subscribe to our group at



Join our community on



Follow us on



www.teatimewithtesters.com

