THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

DECEMBER 2014 | YEAR 4 ISSUE XI

Sign Up For A Free Demo Today

www.teamqualitypro.com

TQP

TeamQualityPro
SEE EVERYTHING

GET A FULL VIEW INTO ALL YOUR DATA

- Inhouse/Offshore contracts
- Defects
- Test phases
- Software Development
- Reports
- Payroll/ Contracts
- All projects
- Notifications

- Project Progress
- Team Analysis
- Top 10 Best / Worst
- Cost Measurement
- Defects / Testing

TAKE THE FIRST STEP IN SAYING:

YES TO:

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

NO TO:

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

testomaton

Quality Assurance via Automation

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

Services

Test System Analysis and work estimation

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

Tests creation and Automation

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

Regression and Test evolution

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

Architecture Consulting

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

Trainings

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

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

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

Another great way to begin your new year is....

IN THROUGH THE SIDE DOOR

A video talk by Michael Larsen

Stay tuned for more updates...

WWW.TVFORTESTERS.COM

TEA-TIME WITH TESTERS

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

Created and Published by:

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

Editorial and Advertising Enquiries:

Email: editor@teatimewithtesters.com

Lalit: (+91) 8275562299

Editorial

The Story of Production (and layoff)

Once upon a time in jungle, there was a Tiger and he had his own factory.

An Ant used to work there. Yes, one single Ant. She used to work as per her own schedule and methods. And used to leave for home after finishing her work, daily. Tiger's business too was running smooth.

One day Tiger asked himself, "If this Ant performs so well without anyone supervising her work, how great she would perform if I appoint someone to supervise her?" And with this purpose, Tiger appointed Honeybee in his factory as Production Manager.

Honeybee had rich experience of her work and she was great at writing reports. She said the tiger, "First of all, we'll need to fix Ant's office timings. And to keep the record of that I would need one secretary." Tiger then appointed one Rabbit as secretory for Honeybee. Tiger was extremely happy to see Honeybee's passion that reflected from her demands. And one day he asked Honeybee to show him all the work done so far and the *graph* of production *progress*. For that, Honeybee demanded a computer, projector and laser printer. Tiger bought all those and he then had to appoint one *Computer Head* to look after all such requirements. He appointed a Cat for this purpose.

Over the period of time, Ant got overloaded with all those form fillings, repetitive report creations, creating so much of metrics, documents over and again. And with all those 'work related' things which were far less important than her actual work. And that in turn adversely impacted her productivity. Production of Tiger's factory got reduced to considerable extent. Then Tiger thought, "I must bring in some technical person who will better explain Honeybee's ideas to Ant in language she understands". And then he appointed one Monkey as *Technical Instructor*.

An Ant, who used to work with her own expert methods earlier didn't like all that new burden, additional processes and she got angrier. Thus, production of Tiger's factory got reduced even more. Realising that he was facing heavy losses, Tiger appointed an Owl to find out the root cause. After three months of survey and investigations, Owl sent his report to Tiger.

The report said, "There is much more staff than needed. You must lay them off". And who went home with pink slip then? Any guesses?

That's correct! That Ant is still in shock, totally clueless of what was her mistake. My special thanks to original author of this story, whoever has written it.

It's 2015 already but testers in our industry at large, are still getting overloaded with non-productive activities, mostly in the name of standards, best practices and creating auditable artefacts. And in the end, **they** are held responsible for lack of productivity. Wish Tiger could know how to do more with less.

With State of Testing Survey 2013, we could find out some interesting facts about our profession and its state last year. It will be interesting to see where we are heading in near future. Please help us figure it out better by participating in it and spreading it to the world.

2015 seems to have started on difficult note for the world but I'm sure that we all will be able to wipe out all bad and we'll progress with all good in. I dedicate this issue to the world-peace and brotherhood.

Wish you all a great new year ahead!

Bhamare

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

Hi Lalit,

I wanted to let you know that I'm one of those readers you wrote about in your editorial: those who can't wait for each issue of TTWT—and who have little patience for late arrivals.

This issue (November 2014), though, was well-worth waiting for. I started with the interview of Markus Gartner. Markus was one of my very best students, and all of his writings are extremely worthwhile. Then I skipped to T. Ashok's article on telling a story, because T. always writes a stimulating article. I totally agree with the premise and reasoning of this essay. I hope it will influence the way testers report their work, but story-telling is not something a person learns overnight.

Vu Lam's article on the three drivers that fuel software success. I thought at first that this was my introduction to Vu Lam, but then I realized that his product, QASymphony incorporates these drivers in an exemplary way.

I then moved onto Klahr's (Anne-Marie Charrett and team) article on discovery experiments, extremely pleased to see someone taking on the task of experimenting about experimenting itself. I was disturbed, however, to see that the work was done in 1988, yet there was no sign of any follow-up in 16 years. The article offers dozens of ways in which future experiments could usefully extend these results. I hope somebody takes up the suggestions.

Wayne Yaddow is obviously an expert in data warehouse testing. I'm sure it's much more informative and interesting when he presents this lecture live.

All in all, a great way to end a great year or start a new great year.

- **Jerry Weinberg**

Dear Lalit,

As the New Year begins, I would like to dedicate this poem composed by me to TTwT and all of its readers. Hope you'll like it.

Everyone walks on the path, we all have to walk
but my feet have learned to walk on the unbeaten paths

You may laugh on me, please but I'll remain determined,
Everyone has their own way of living life...

Paths which are already lead by someone,
My feet don't like walking on them...

I might face the thorns & stings but I'll create my own path,
You may please keep dreaming of paths full of blossoms & roses...

I'm like a swirl, I want to go deeper in this sea,
You may please sit on the bank & enjoy talking with waves...

I'm not the first one to dive deep to find pearls of wisdom,
But how many were there before me who tried to do that?

I am too a wonderer but only difference between me & them is, I am in a search of that invaluable pearl...which has not been found yet by them…& I'm crazy for it...

You may please remain happy with things found with ease
I'm going far away...I want to go deeper...

You may please....You may please...

- **Dr. Sanjay Gupta, Dell Services - Bangalore**

We love your feedback.

Write to us on editor@teatimewithtesters.com

TEST ISTANBUL 2015

www.testistanbul.org

PERFORMANCE TESTING: HIGH PERFORMANCE SOFTWARE DRIVEN BY BUSINESS

27 March 2015

Software Testing Conference

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

Keynote Speakers

Goranka Bjedov
facebook

Alexander Podelko
ORACLE

Martin Spier
NETFLIX

Ian Molyneaux
intechnica

Super Early Bird Fee
Valid until 8 December 2014
270 EUR *VAT excluded*

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

ISTQB International Software Testing Qualifications Board

TTB Turkish Testing Board www.turkishtestingboard.org

Event Partner:

keytorc

Supportive Organizations:

PROFESSIONAL TESTER

BETTER SOFTWARE

Tea-time with Testers

QuickLook

STM
NOVEMBER - 2014
YEAR I ~ ISSUE I
SOFTWARE TOOLS MAGAZINE
© Copyright 2014. Tea-time with Testers. All Rights Reserved.

TQP

QASymphony

SmartBear SOFTWARE

The Top Five Companies for Job-seeking QA Engineers

The technology sector dominated The Huffington Post's 2014 list of the happiest jobs in the United States, with quality assurance engineers coming in second. Quality assurance engineers, whose job is to ensure all programs are working properly, are generally hired by larger companies that produce software or are in web development. QA engineers, even at entry-level, make well-above the national salary average, and they report positive job security and growth opportunities.

All QA jobs; however, do not guarantee personal satisfaction. Depending on an individual's preferences, QA jobs can range from working in large-scale corporate headquarters in New York City to fresh new start-ups in San Francisco. Below is a list of the top five hottest companies for job-seeking QA engineers, including how to find and land the gig.

1. Google - Google offers employees excellent compensation, full benefit packages, job security-- and prestige. Of course, in order to get these perks, which include paid maternity leave, travel insurance, and on-site health staff, you have to pass through five rounds of interviews. According to William Poundstone, author of Are You Smart Enough to Work at Google?, the interview process is designed to assess an applicant's' knowledge in their field of expertise, their ability to offer creative solutions, and their "fit" within Google's extroverted employee landscape. Google jobs can be found on their career website. There are currently several QA positions available, both at their headquarters in Mountain View, CA and via telecommuting.

2. Apple - The majority of prospective Apple QA engineers found their jobs by applying directly online or through employee referral. The interview process consists of several short phone interviews, followed by a full day on-site interview with various staff members. Once hired, Apple employees report high levels of job satisfaction, largely thanks to a positive work environment (that feels more like a start-up than a corporation), unprecedented employee benefits, and the knowledge that their work has an impact on the world. According to career website Glassdoor, the average salary for an Apple QA engineer is approximately $93,000 – a number well above the industry standard.

3. QASymphony - A comparatively smaller start-up, QASymphony, is a test management platform that helps companies check their products for software bugs before launch. The QASymphony team is a close-knit group of software developers, IT professionals, and information revolutionaries, all working together to create innovative products for bug testing and tracking. Based in Atlanta, QASymphony currently serves over 604 companies in 78 countries.

For those who prefer to be a big fish in a small pond, start-ups such as QASymphony offer employees a varied, individualized work experience. For open roles such as being a Software Engineer, visit QASymphony’s Linkedin page.

4. Airbnb - If working at a large corporation is not the right fit for you, consider well-established start-ups like Airbnb. A now well-known name globally, Airbnb is valued at approximately $10 billion although it was just founded in 2008. Headquartered in San Francisco, this online community rents private and shared spaces internationally. Engineers at Airbnb “own their own impact,” meaning that each engineer is responsible for creating his own company value. This individual responsibility is balanced by a default to information sharing and an emphasis on structured teamwork. If you’re seeking employment at Airbnb, make sure you possess Pixelwax: their term for the dedication and craftsmanship that all employees must bring to their work.

5. Lockheed Martin - For those interested in careers in the governmental sector, Lockheed Martin, a global security and information technology company, may offer the solution. The majority of this company's business is with the U.S. Department of Defense and U.S. Federal government agencies, making Lockheed Martin the largest provider of IT services and training to the U.S. government. According to LinkedIn, there are currently 94 QA engineering jobs available across Lockheed Martin, with locations ranging from Fort Worth, TX to air force bases in California and New Mexico. With a 2013 sales revenue of $45.4 billion and a backlog of $82.6 million, this corporation was ranked 59th on Fortune 500's 2014 list.

Press Contact:

Marisa Negri

Business Analyst, Techwood Consulting

A million dollar smile ?

Ask our <u>sales team</u> about
our **Smiling Customer** ☺ programme

*Adverts starting from $100 USD | *Conditions Apply

Photo credits: Amir Eltahan

He is well known in the Context-Driven Testing Community. His way of testing is influenced, developed and inspired from and by James Bach, Michael Bolton, Cem Kaner, Scott Barber, Jerry Weinberg, and others. He can offer you state of the art testing to match your business needs in the most efficient way. He provides leadership and management consulting and coaching. He tests, coaches, consults, speaks, writes, manages and thinks about software testing and problem solving.

Meet my friend **Henrik Andersson**, one of the passionate Context Driven Testers and activists I have met in person. Find out what all we talked about, in this exclusive interview.

- Lalitkumar Bhamare

Over a Cup of Tea with Henrik Andersson

It’s pleasure to be talking to you today, Henrik. You are already well known within Context Driven Testing community. Would you like to share your test side story with us?

I'm Henrik (many knows me as Henke) one of the many great testers in Sweden. This answer could be very long, there are so many things that have happened that I’m really proud of but instead of a long rant I spread my answer out over the upcoming questions.

I have been involved in testing since the late nineties and over these years in a wide verity of businesses and roles.

My entry into the CDT community happened around 2007 when I invited James Bach to Sweden to do his Rapid Software Testing course. This was the first time I met James and I had the opportunity to spend time with him. At first I got the third degree interrogation but that turned quite quickly into some really nice and colorful discussions. Before James left Sweden he told me that just had to come to Seattle a few month later to be part of the peer conference WHET and the CAST conference. I told him that would be tricky to get my tight-fisted employer to agree to that spending. But James talked to his brother Jon who was the chair of both WHET and CAST. Jon kindly sent me a personal invitation email to come and participate and that did the trick to let me go. In Seattle I got introduced to a wonderful community of testers that I connected very well with. From there on I have part of the CDT community and I’m very thankful to James and Jon for this kind introduction to the community.

It did not take long until Jerry Weinberg’s name popped up. I started to read his books and at CAST 2008 in Toronto I had the opportunity to attend one of Jerry’s workshops. The year after I took the Problem Solving Leadership training (PSL). I have been a returning participant at Amplify Your Effectiveness conference (AYE) and last year his latest Change Artistry. Jerry and the whole AYE crew have had a profound impact on my work and how I operate.

What motivates you be a tester?

It was quite some time ago I worked directly with testing. Today I’m running a company, the Lets Test conferences, ISST (International Society for Software Testing) and ConTest; our local CDT meet up in Malmö Sweden.

There is a thread in what I do and it is building community of skilled testers. This is what motivates me to be in this craft. I’m devoted to help moving testing forward and helping us all to grow our understanding of testing. For me this is incredibly rewarding and gives me the opportunity to meet testers all over the world. As much as everyone wants to be unique we still have much in common and facing the same challenges. I believe that there is so much we could achieve if we just step out of our own little bubble and reached out to others

In your career so far, you have worked at almost all roles within testing space. Which one out those did you find most challenging and why?

One huge challenge that I and many with me are facing is getting organizations to recognize skilled testers. To stay away from the low prize commodity testing services or in recruitment not to require a shallow ISTQB certification as a proof of testing skills but instead start to challenge candidates and require them to demonstrate their skills and how they can contribute to an organization.

Would it be too much to ask that recruiters and managers started to do their damn job when hiring testers? If your checklist consists of having an ISTQB certification, specific technical knowledge, specific domain knowledge and of course the person shall work in a team so he/she needs a half decent personality too then you are not doing your job. It’s not something I’m making up, this is one of the most common specifications I see in Sweden that companies base their recruitments on. Of course technical and domain knowledge is good to have but you know what, this just does not cut it. Please stop wasting your employer’s money by hiring incompetent testers, I bet if it was you own money you would be much more careful. Your job when hiring a tester is to figure out what your team really needs to solve the tasks and challengers. You need to be qualified to actually evaluate the tester’s skills and capabilities. You, as a recruiter need to be able to see through the shallow textbook answers and dig deeper in to the testers capabilities in reasoning, strategizing, critical thinking, lateral thinking, describing, risk awareness, questioning, storytelling, modeling, curiosity and all that you’d need to value for your investment.

... CONTINUED ON PAGE 44

The Three Great Obstacles to Innovation

NO-PROBLEM SYNDROME: THE NUMBER TWO OBSTACLE

On the same trip that we visited Shirley, I encountered an example of the second obstacle that interferes with every effort to become a more effective Problem solver. I call it the No-Problem Syndrome. I was in Sacramento, addressing the local chapter of the Data Processing Management Association. I began my address by talking about a previous trip as a new employee at IBM, a visit I'll never forget.

The State Legislature had just passed a law allowing letters as well as numbers on license plates. Opponents of the law had argued that certain combinations of letters might prove offensive. The bill's backers promised to cull all offensive letter combinations from the plates, but they had no particular plan as to how to go about this. Somebody told them that a computer would be a big help, so they called IBM. That's where I came in.

I was a fresh young IBMer, all suited up and ready to purge the world of dirty words with a whiz-bang computer program. Unfortunately, the people from Motor Vehicle Registration had at least three requirements that I couldn't possibly satisfy:

• Some "offensive" words weren't words in English but only looked like words in English. To understand this problem, drop one letter from your favorite four-letter expletive. Sometimes the new word is innocuous, but other times it's as offensive as the original.

• California had many ethnic groups speaking many different languages. The program was supposed to get rid of anything that might be offensive to anybody speaking Spanish, Chinese, Hebrew, Yiddish, Greek, French, Armenian, and a few others I can't remember.

• We also had to remove words that might someday be offensive to anybody speaking any of these languages or any languages of anyone who might someday visit California.

I told the license plate story to kick off a discussion of what makes problems difficult to solve. Then I distributed a set of problems for everyone to try. As I circulated through the room to see how everyone was doing, I noticed one man sitting conspicuously with his arms folded tightly across his chest.

"Have you finished already?" I asked him.

"No," he said, "I'm not doing them. Why waste my time? Why don't you just hurry up and tell us what it is you're trying to tell us?"

"I can't tell you," I said, "because I want you to feel the frustration of trying to do certain kinds of difficult problems. Telling you just isn't the same."

"Well, you might as well tell me," he countered, "because I can solve any problem you can give me. In fact, your 'unsolvable' license problem is actually trivial."

"Trivial?" I asked.

"Absolutely. With modern technology, all you need is a big dictionary. You pass the combinations of letters against the dictionary and eliminate the ones you don't want. It's just no problem."

My first instinct was to argue. I might have asked him how he was going to get a dictionary of words that hadn't been coined yet. Or why you would bother with a computer, once you had constructed a dictionary of offensive words. Then I realized that the poor man was suffering from a severe case of "No-Problem Syndrome," or NPS for short. I had suffered from this disease myself, so I had nothing but sympathy for the poor man. I don't get any kicks out of attacking the handicapped, so I simply smiled and walked away.

Perhaps you've never heard of NPS? I haven't checked this with any neurophysiologist, but it seems to be a condition in which the ears are not properly connected to the brain. The sounds enter all right, but they trigger a stereotyped response that has nothing to do with their meaning. One person describes a terribly vexing problem, but the other merely responds with a callous, "No problem."

No-Problem Syndrome isn't the same as deafness. In fact, deaf people couldn't have NPS because the response has to be triggered by the key word, "problem," reaching the ear. Once that word registers, the ears become selectively deaf, the first stage of the syndrome. The second stage seems to be the mental retrieval of some favorite solution method, which is immediately presented to the talker, even if it's necessary to interrupt the problem description. My Uncle Max had NPS, and his favorite solution had to do with restoring the practice of beating children in the public schools, much as he beat his children at home. If the economy was down, it was because they don't beat kids in school any more. If crime was up, or the weather was bad, it was for the same reason.

Like most kids, I didn't understand the heartbreak of NPS. I used to laugh at my uncle, never dreaming that NPS might be hereditary. Because I lacked self-awareness, it wasn't until after my first visit to Sacramento that I found out the tragic news: I had NPS myself!

It was the license plate problem that revealed my terrible secret. The Motor Vehicle people told me about their requirements, but evidently their words never registered in my brain. Before they had finished talking, I had assured them that there was no problem. Before I understood what their problem was, I had managed to write a program to solve it.

You can imagine what a blow it was to a young IBMer when a bunch of civil servants rejected all his brilliant work. On the other hand, you can imagine what a blow it was to them when a young IBMer, not even listening to their requirements, told them they didn't really have a 'problem. It may be terrible to have NPS, but it's even worse to be one of its victims.

I used to think that computers emit some nerve-damaging high frequency sounds, because NPS seems to affect a large percentage of computer professionals. Whenever they hear the words "problem" and "computer" in the same sentence, they launch into a diatribe that would put old Uncle Max to shame. And it always starts with the words, "No problem."

As I grew older, however, I realized that computers themselves do not actually cause NPS. Perhaps it's the fast pace of the industry, which reduces the likelihood that problem solvers will take time to be aware of what they're doing. Besides, I've noticed that NPS afflicts people in all high-tech industries, so it can't just be computers. It also affects quite a few people in low-tech industries, or people not in industries at all.

My own symptoms have abated a bit with age, so perhaps the problem is simply a manifestation of the youthfulness of the computer industry. Other than senility, though, I don't know of any cure for NPS. I wish I could help these poor handicapped souls, but bitter experience has taught me that I'd have more success going to Sri Lanka to cure lepers.

People who have problems for others to solve should be better informed about the perils of NPS. Because it can't be cured, they had better learn to protect themselves through my four-step plan for early NPS detection:

1. You describe your very difficult problem.

2. The respondent says, "No problem!"

3. You say, "Oh, that's terrific! Could you please describe my problem that you're going to solve?"

4a. If the respondent then describes your problem, even erroneously, it's not a case of NPS but only a case of Enthusiasm.

4b. If the respondent describes a proposed solution to your problem rather than the problem itself, then sadly its NPS. The kindest thing you can do for all concerned is smile and walk briskly to the nearest exit.

Sometimes the four-step detection plan can be used for self- diagnosis, but if the NPS is too far advanced, it won't work. To detect your own NPS, you have to be able to hear yourself say, "No problem!" or at least hear yourself giving solutions before you've confirmed that you understand the other person's problem. But, alas, terminal NPS patients can't hear other people very well; and they can't hear themselves at all. They're not only self-blind, they're self-deaf.

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.

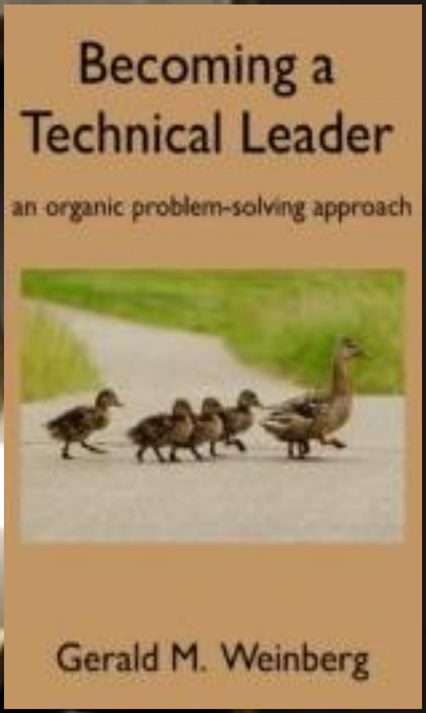
For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including The Psychology of Computer Programming. His books cover all phases of the software life-cycle. They include Exploring Requirements, Rethinking Systems Analysis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

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

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

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


Becoming a Technical Leader
an organic problem-solving approach
Gerald M. Weinberg

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

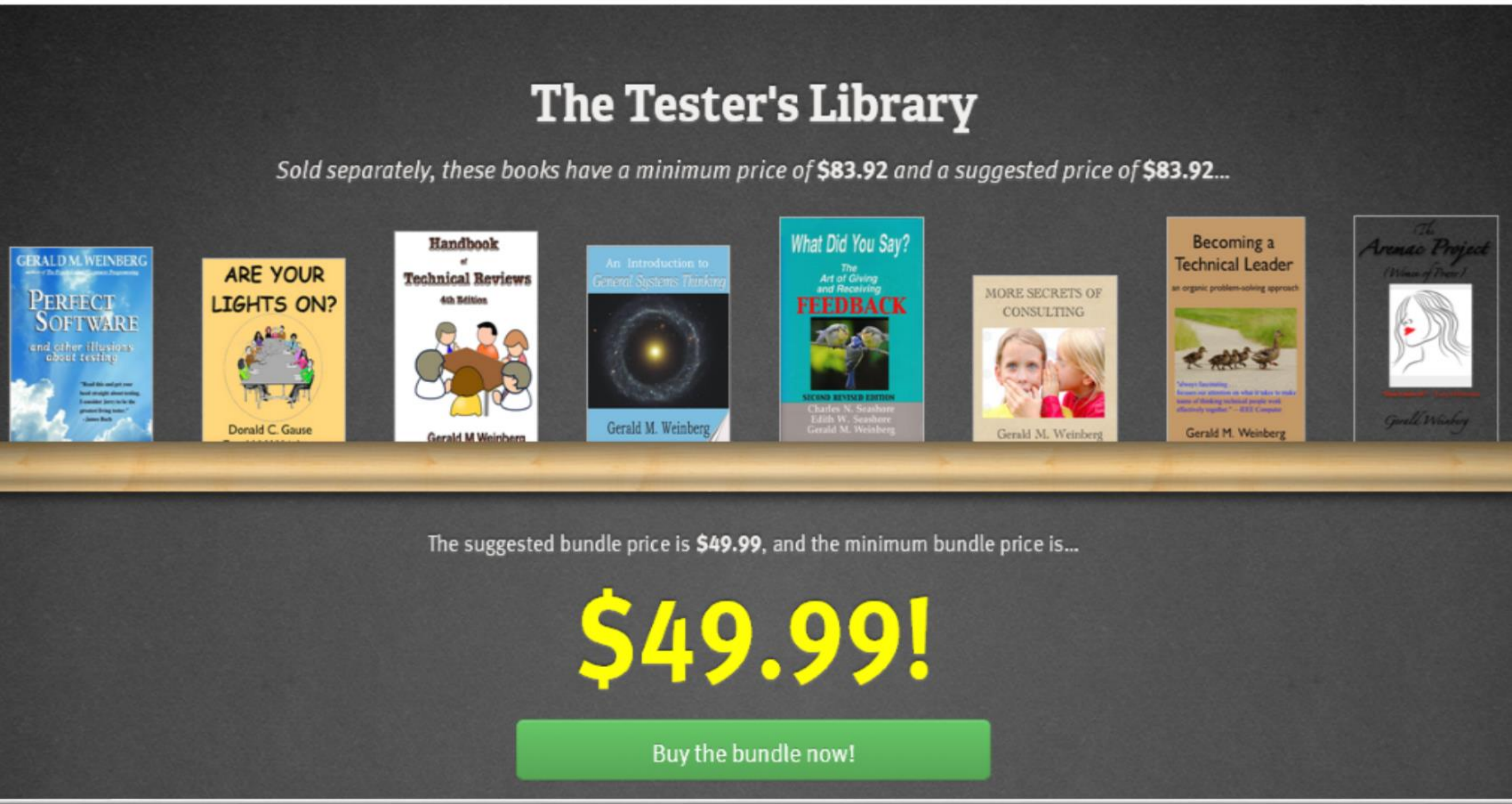
TTWT Rating: ★★★★★

Its sample can be read online.

Know more about Jerry's writing on software on his website.

The Bundle of Bliss

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



The Tester's Library

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

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

$49.99!

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

The 8 books are as follows:

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

Know more about this bundle

TEA-TIME WITH SMARTBEAR

About this column…

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

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

API Testing – Do or Don’t You?

- by **Lorinda Brandon**

Why are APIs so important for web application testers?

With today’s emphasis on mobile apps and web applications, there is an increased reliance on APIs to power the communication and data exchange between client and server. In addition, many developers rely on third-party APIs to provide capabilities they don’t want to build themselves, like payment engines, maps and directions, and shipment tracking. It’s no wonder Forbes has coined this “the API economy”. APIs have become more than just technical glue; they often have ties to a company’s bottom line because they enable partnerships and brand awareness with an ease we’ve never seen before in this industry.

If you test web applications or mobile applications, you are also testing APIs, whether you know it or not.

API Testing Strategies

For web application testers, the question really is whether you are actively testing those APIs or passively testing them. Many testers concentrate on the application layer itself and test the API only by uncovering aspects of the application that are failing because of the underlying API. While this can work, it doesn’t offer much focus to what is arguably the backbone of any great application – its API. Passive API testing can fall short in many ways so it’s important to have a strategy that lets you uncover issues early and troubleshoot effectively.

Test Coverage

By exercising the API passively through the application, you are not fully exploring the capabilities of the API as designed. Most applications only leverage the aspects of the API that are needed for a specific transaction; however, as the app evolves over time and its use of the API is modified, you can encounter issues with the API that you could have managed much earlier in the cycle. By using the API definition as your test skeleton, you can ensure that you are fully traversing all of the available methods.

Load Test Isolation

Any load tester will tell you that the challenge of analyzing the load test results is following the code path and the network connections to find the performance bottlenecks. When you get application load test results, how can you determine whether the bottleneck is at the API level or within your application or databases? Without discrete results for the API, you also can’t determine where the problem is within the API – is it one method causing all the issues? Separating your application load tests from your API load tests will help you isolate the results of each so you can troubleshoot any issues more efficiently.

Environment Testing

If your application relies on an API that is impacted by slow connection speeds or underpowered servers, your application will also exhibit that same impact. It’s important to know the effect of a poorly performing API on your application so you can ensure your user’s experience is not impacted. Simulating a variety of error conditions and server limitations using a virtual service can help you understand the API's behavior under those conditions so you can see how your application responds.

Parallel Testing

Often the API your application depends on is being built or modified at the same time as the application. How then do you test your application reliably if it depends on an API that is still in development? One important aspect of understanding your APIs is having a service description. There are a lot of formats available and it doesn't matter which one you favor – any of them will help you determine how deep and wide your test path needs to be. You can also use the service description to generate a virtual service that you can use in your testing without disrupting development. Parallel testing means taking iterative builds as you go so you can stay current with development – look for an API testing tool that can automatically refactor your tests when the API changes so you don't spend all your time rewriting your tests.

Third-party API Testing

Many applications today rely not only on APIs built in-house but also on APIs provided by a third-party (easy examples are Twitter, Facebook, and Google Maps). While this provides a faster way to build functionality from a developer's standpoint, it presents some challenges for testers because you cannot control or impact the quality of these APIs. While many of the testing strategies outlined above can help with these challenges, there are some other considerations to take into account as well.

Testing the Application, not the API

With third-party APIs, it's even more critical to separate your application testing from your API testing. You don't control the quality or development of a third-party API but you do control how your application responds to API failures. When you plan your testing, make sure the focus of your testing is on your application's behavior in relation to the third-party APIs you rely on.

Using Virtual Services for Simulation

There are a lot of good reasons for using virtual services to replace any third-party APIs you depend on. Many third-party APIs charge for use, either for the amount of data transferred or the number of requests, and this is an unnecessary expense for testing activities. By using a virtual service, you can avoid those charges and free yourself from any expenditures that might hamper or limit your testing. But the big benefit to virtual services in testing is being able to simulate a variety of environment limitations (low bandwidth, limited network, etc.) and error conditions that might impact your application's ability to function. With APIs being so core to the application's basic functionality, being able to simulate suboptimal situations is a huge advantage to a tester.

It Doesn't Have to Be Complicated

API testing can sound like a daunting prospect but it doesn't have to be complicated. Start with a clear description of what your API does and how your application uses it, then build from there to develop tests than can isolate your API failures from your application failures.

Happy Testing!

Tea, Testing and QASymphony

Measuring Quality in the Post Zero-Bug World

Agile development has undeniably shifted the focus on quality away from the waterfall method of creating software that ships with zero bugs, to creating software that provides meaningful value to the end user.

Before I get excoriated by the testing community on twitter, please note that these two things can coexist, it's just that our means to arrive at quality has changed. Let me explain:

In the old "Zero-Bug" world, testers tested every preconceived use case of the software that could be imagined at the time of design. Testers created test plans in painstaking detail to test every possible "what if" scenario that could occur along the way. Well-known testing and development shops often prided themselves on metrics such as 100% pass rates and 0 known bugs.

You might think "well then, why would anyone move to Agile?"

That's easy - because bugs only paint one part of the picture; the part you defined when you began to test and develop your software. Often, I've found that many of these heralded "Zero-Bug" applications have "feature request" lists that would intimidate even the most stalwart Product Managers. The thing is, it's easy to dismiss user feedback as a "feature request" and claim the software works as designed - especially when the same team that designed the software tested it. But what does that really mean if neither the tester nor designer is the one actually using it?

Therein lies the beauty of agile. We throw these semantics out the window and focus on what really matters, which is delivering value to the actual end user. Instead of trying to test everything to answer the question "what if", we add some perspective to our efforts and ask "why then". There is no undue importance placed on "bugs" vs. "feature requests", and the count of these may be tracked but it is not taken as the primary measure of quality.

So how then can we measure the quality of our software in this Agile age then?

The answer to that question is neither simple nor unilateral, and anyone who tells you that it is should be questioned. That's because quality in an Agile world is dependent on value, and value is derived from the customer. Since it would be unlikely that two products would share the exact same customer base, it would be unlikely that two organizations would share the exact same quality measures.

I would urge all agile quality teams to think outside the box and interact closely with their customers to understand what really drives value for them, and how metrics that track that derived value can complement traditional quality metrics. Some measures I have seen used to great success in organizations include (but are not limited to):

- User Engagement (using Intercom.io, kissmetrics, etc.)
- Net Promoter Score
- Social Media mentions
- Referral business %

What you will notice about these metrics is that they are not tied only to testing activity, but rather shared across many business units. This is aligned with the idea that in an agile development organization, everyone is involved in quality to some extent, testers are just the champions for it.

Kevin Dunne is a Product Specialist at QASymphony where he manages support operations for 10,000+ customers across 600+ companies and 78 countries, including live support, online ticketing, documentation, and training.

You can find out more at www.QASymphony.com

Back To Index

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

Can playing games improve tester skills?

- by Christin Wiedemann

Exploring the science behind games

Introduction

Are gamers predisposed to careers in software testing? Is there something in the mindset of gamers that make them especially good testers? The prevalent perception seems to be that testers enjoy playing games more than the general population, and that playing games makes us better testers by honing specific cognitive skills considered especially important in our field. Can it be that people who enjoy games, riddles and puzzles are indeed better equipped to handle challenging software testing tasks?

Reading blogs and tweets, and listening to presentations and discussions at conferences, there is a hypothesis – often treated as a theory – that playing games is an important part of gaining and improving tester skills, but there is a disconcerting lack of empirical evidence accompanying such claims. Our job as testers is to question unsupported statements and to put hypotheses to the test; therefore I decided to go on a quest to discover if there is any scientific evidence that playing games can improve tester skills. But what do I mean by "scientific evidence"?

The Scientific Approach

My personal belief is that testers should adopt a scientific approach to testing. The ultimate goal of all sciences is knowledge and, to acquire new knowledge, scientists make observations and analyze data – activities we normally refer to as research. For a method to be considered scientific, it must be based on gathering empirical evidence. As testers, we test software to try to learn how it works; like a scientist, our goal is to gain new knowledge. When we test software, we are in fact experimenting and observing the results. Testing is simply gathering empirical evidence.

Simplified, the scientific method involves the following workflow:

1. Collect data through observation
2. Propose a hypothesis and make predictions based on that hypothesis
3. Run experiments to corroborate the hypothesis

If the experiments corroborate the hypothesis, additional predictions can then be made and tested. If the experiments instead refute the hypothesis, it is necessary to go back and propose a new hypothesis, given the additional knowledge gained from the experiment.

For me to accept the claim that playing games makes us better testers, I needed to see empirical evidence, derived using the scientific method.

Defining Games and Play

"Play" and "games" mean very different things to different people, and "play" especially is an emotionally loaded word. Play is generally thought of as a voluntary activity that is fun and where the primary purpose is not to achieve a specific goal. Unfortunately, play is often contrasted against work, but play is important in the work environment too. Play encourages teamwork, helps us build relationships and promotes creativity and innovation. Play is a way to learn and grow, both as individuals and teams, and should in itself never be considered a waste of time. [Reference: "Play: How it Shapes the Brain, Opens the Imagination, and Invigorates the Soul", Stuart Brown, Penguin Group, 2009] When I use the phrase "playing games" in this article, my definition of "games" is very broad, including board games, puzzle games, riddles, video games, etc.

Before trying to answer if there is any scientific evidence suggesting that playing certain games can improve tester skills, we need to break down the question and look at one part at the time:

1. Can cognitive skills be improved at all, regardless of method?
2. Can playing games improve cognitive skills?

Neuroplasticity

Historically, the brain has been seen as static, a hardwired computer whose circuits are finalized in our childhood, but it turns out that the brain is anything but static – the brain is plastic and can be rewired. Neuroplasticity refers to changes in neural pathways and synapses due to changes in behavior, environment and neural processes. It is possible to change both the anatomy (the structure) and the physiology (the functional organization) of the brain. In other words, science indicates that at least the brain can change, which means that it is plausible that cognitive skills can be improved. But can playing games change the brain?

Current Research

As a layman, gaining insight in to today's research on the relationship between playing games and neural development is not easy. Most studies are published in journals that the public does not have access to, and what appears in public media and easily available literature tends to be skewed and sensational.

There are also a lot of readily available studies done by the very companies that make a livelihood out of providing commercial brain-training games. It's hard to believe that these studies follow the scientific method and are unbiased.

To make matters worse, setting up an experiment to determine if playing games improves cognitive skills is far from trivial. How do you measure the baseline, i.e. the skill level before playing the game? If you use the same test, or type of test, to measure skill levels before and after playing the game, then how do you distinguish improvements caused by the subject learning the test from improvements caused by playing the game? There are also studies on children that are quite long, which makes it hard to know if the improvements seen are from playing the game, or just natural learning that could happen in six or twelve months. Additionally, a lot of studies do not use a control group, and how do you design a control group for, as an example, a study on the results of playing Mastermind? What do you have the control group do? Finally, there is the problem of sample sizes. Many studies use groups of just a dozen or so people and, as a consequence, the statistic reliability of the results is very low.

Equipped with my scientific outlook and skeptical thinking, I took a deeper dive into a few of the more interesting articles I found. I will be the first to acknowledge that the sample I selected is not representative of the current research in the field, but randomly picked.

Reasoning Skills and Speed Training

I found one study [Reference: Developmental Science 14:3 (2011), pp 582–590] on children's learning that looked at reasoning skills and processing speed. Reasoning skills account for our ability to plan and build new relations between elements, and processing speed measures rapidness of visual detection. Reasoning represents the capacity to think logically and solve problems in novel situations, which is a very important skill for testers that are constantly faced with new and challenging situations and software behaviours we have not seen before. Processing speed corresponds to our ability to quickly process inputs and, in particular, perform quick, visual searches, something tester are frequently doing while working with a GUI or checking log files.

In the study, a group of 7-9 year olds played games for two hours per week for eight weeks. One group played twelve different reasoning games, computerized and non-computerized, individual and collaborative. The second group played twelve speed of processing games.

The children's cognitive skills showed large improvements, and the group that played reasoning games only improved their reasoning skills, whereas the group that played speed of processing games only improved their processing speed, which shows that the processes are independent. The results of the study implies that reasoning and processing speed can be improved, at least in children, and those skills are important to testers. But is there a lasting effect, and does it transfer to adults? A lot of the research I came across turned out to be focused on children's learning, but I did find an interesting experiment that was focused on adults – Brain Test Britain.

Brain Test Britain

The Brain Test Britain experiment [Reference: https://ssl.bbc.co.uk/labuk/experiments/braintestbritain/] was conducted through Lab UK [Reference: https://ssl.bbc.co.uk/labuk/], which is a BBC website that encourages citizen science and invites the public to participate in groundbreaking scientific experiments. The Brain Test Britain experiment was a full clinical study launched in 2009, designed by researchers at the University of Cambridge and Kings College in London.

13,000 people spent ten minutes three times a week for six weeks playing brain-training games. The participants were split into three groups:

- One group playing reasoning games
- One group playing non-reasoning games
- One control group

This study only looked at computer-based games, and the control group did tasks that involved using the Internet but not any actual brain training.

The study found no evidence that playing brain training games transfers to other brain skills. “Practice makes perfect” – playing a specific game makes you better at…that particular game, but playing games doesn’t make your smarter, or boost your brain power.

Conclusions

I looked at additional studies with completely different setups, using both computerized and non-computerized games, with people playing games both individually and collaboratively. There is definitely scientific support for the idea that playing games can improve cognitive skills such as:

- Procedural skills
- Debugging skills
- Visual search skills
- Attention skills
- Reasoning skills
- Processing speed

However, the transfer of skills out of the context in which they were acquired appears to be limited or even non-existing. Playing a specific game makes you better at that particular game but, based on the information I gathered, I would not claim that playing games improves cognitive skills per se. The experimental setups and hence the reliability of the studies can also be questioned. Nonetheless, there seems to be no doubt that the brain can change.

So, is there any point in playing games or should we stop playing? Whether playing games makes us better testers or not, there are other benefits to consider too. First of all, it’s fun! It can provide stress relief and an opportunity to de-focus. I also believe that games can be used to provide a safe environment in which people can learn the value of being skeptical and questioning.

I set out to decide if there was any empirical evidence that playing games makes us better testers, and even though I think the information I found was inconclusive, I will most certainly continue playing games and encourage my fellow testers to do the same.

After finishing her PhD in Physics at Stockholm University, **Christin Wiedemann** started working as a software developer for the Swedish consulting company HiQ. Christin soon discovered that software testing was more interesting and challenging than development and subsequently joined the Swedish test company AddQ Consulting. At AddQ, she worked as a tester, test lead and trainer, giving courses on agile testing, test design and exploratory testing throughout Europe. Christin developed a course on exploratory testing, and is a co-creator of the exploratory testing approach xBTM. Christin currently lives in Vancouver, where she joined Professional Quality Assurance (PQA) Ltd. in 2011. In her current role as Chief Scientist, she drives PQA's research and method development work. She continues to use her scientific background and pedagogic abilities to develop her own skills and those of others.

Over the years Christin has been fortunate to work together with many great testers, including **Martin Hynie** who originally came up with the idea that lead to this article. For over a year Christin and Martin have worked together, exploring the science behind playing games, presenting continuously updated results at test-themed conferences.

Why Challenge Requirements?

- by Llyr Wyn Jones

I know I've lead this article with "requirements", but please bear with me.

Over the past few months I have been researching and reviewing a plethora of test case design methodologies, specifically formal modelling techniques, and a disturbing thought dawned on me: most informal test case design methodologies assume that the requirements are perfect (or at least correct). There are scant examples of methods that actively challenge the quality of requirements - this was a very strange notion to me (as a developer and mathematician, I had assumed that most software development would be done with at least a modicum of engineering rigidity).

Essentially, I'm going to explain, using a couple of situations I have found myself in as a consultant, why challenging the requirements is a good thing, both from a short- and long-term perspective. Even in a waterfall environment (Agile simply does not have an excuse not to do this) where silos are the norm, it can yield many benefits.

Challenging Requirements

Let me first explain what I mean by "challenge", using two real-life scenarios:

1. **Test data requirements from a tester to a member of the test data team.** The requirement itself is vague, horribly worded, and the person provisioning the data has absolutely no clue how to proceed.

   In this situation, the bad requirements are a bottleneck, and we advised the data team to challenge the requirements and only service ones which were fully specific. It was interesting, to say the least, to see the improvements in their requirements in the following weeks: within a month, the throughput of the data team was much improved simply by virtue of eliminating the bottleneck of bad requests clogging up the system.

2. **A tester attempts to build a functional test suite for a rules engine**, and the requirements themselves, on the face of it, appear to be fully specified and detailed (personally, these were the best requirements I had, and have ever since, seen). The rules themselves were a plethora of ands, ors and nots, so we used a method called "Cause and Effect Graphing" to logically model the requirement and automatically build a set of test cases for 100% functional coverage. Except, at this point, a glaring hole was found in the requirement - but it was only by reviewing the test cases (and finding a contradiction in two of the 15 test cases) did we detect the problem. It turned out it was an implicit assumption based on language, and was so obvious no-one had thought to test it! We went back to the requirement writers with our findings, and within a day or so they had fixed their mistake and plugged a lot of bugs in the process (including a couple of Severity Ones that were a direct consequence of the error). Again, I maintain that these were the best set of requirements I have ever laid my eyes on: which goes to show that even requirements that look good can still break under logical analysis.

Logical Analysis of Requirements

I, myself, am a practitioner of requirements-based testing (RBT), which, as in the second example above, involves bringing in testing earlier in the cycle by testing the requirements themselves. There are three main tools at our disposal:

1. **Ambiguity analysis** - a deep (and sometimes brutal) tear-down of the requirement in order to determine if there are multiple interpretations. Language use is analyzed, and the resultant unambiguous version is extremely terse and specific. In particular, it is nailed down so tight that even a talented lawyer would struggle to work around it.

2. **Use case/Test case review** - once the requirements have been proven to be unambiguous, the next step is to work out if they are correct. We generally use either flowcharts or Cause and Effect diagrams to represent the logic, and from the diagrams we derive a set of paths - these are our use cases and test cases. Since these can be generated automatically from the diagram, we then review with a subject matter expert (SME) to make sure that they are correct.

3. **Predicate calculus** - in particular, for cause and effect, this technique is used to automatically verify that the requirement does not contain any contradictions. This is how we identified the killer contradiction in my second example above.

By making sure that the requirements are testable, we make sure that the overall project is testable - and if a project is testable, then it can be tested to within an inch of its life, yielding a much superior end product - not to mention high morale in the testing team! As I mentioned before, Waterfall has its issues, and the scope for people who know their stuff to test (or even challenge) the requirements is one of the key ones. However, I have seen requirements in Agile projects which are truly shocking - far worse than any Waterfall requirements I've come across - which is not intuitive since the whole concept yields itself to testing every single step of the SDLC. One particular bone I like to pick is how the Agile manifesto yields itself as an excuse to not do documentation, but that's one I will leave to its own article.

Does it pay?

Now, the question that comes next is usually: "But this will take time and cost money - why would we do this?" Sadly, the fact that testers seem to waste their working hours trying to pick apart long walls of text (usually in the dreaded/beloved Word document) does not occur to management, and the fact that this (frankly, wasted) time can be replaced with a methodology that will improve requirements, improve quality, and will not be as soul-destroying to the testers, does not cross their mind at all. All they see is the up-front investment, and not the long-term benefits (and overall cost reduction).

However, there is another problem. As we all know, trying to get business analysts (BAs) to give us requirements in a form testers can work with is very difficult at best. BAs, in the main, are generally non-technical and rely on what I would call "liberal-arts forms" of specifying requirements: namely, Word documents, Excel spreadsheets and Visio diagrams. These are not ideal, and the reasons go much further than this short list:

1. Non-reusable - all of the above have to be re-factored if they are to be used for any other purpose. For example, Word documents have to be analyzed by a human to derive test cases, whereas a proper tool can derive the test cases programmatically.

2. Tedious - on a flight to Florida for an assignment, I once read through an entire requirements documents for a large project whose requirements I would be analyzing. It took me around 8 hours - and for the most part, I was none the wiser as to what it was supposed to be doing. Apart from the fact that I'm a non-SME, a clearer, lighter form (such as flowcharts) would be much easier to ingest and to understand.

3. Ambiguous - the litmus test I apply is: "can a non-SME understand this?" If not, then it is ambiguous, end of discussion. It should not be necessary to be an SME to understand a project requirement - the alternative is to spend inordinate amounts of time and money training people up on material that they should just simply be able to pick up. But that is not their fault: non-tedious requirements should be very easily ingestible, in bite-sized chunks, without having to read a 400-page wall of text.

I would like to conclude by stressing that this is not an attack on the testing community; rather, it is very much the amalgamation of the many frustrations and concerns shared with me by exasperated testers whose job is made more difficult by having to assemble an understanding of the system from various sources. In the software industry, I feel that requirements, by and large, are not fit for purpose, and it is testers who bear the most of the consequence since it is their responsibility to assure that it all works. Poor requirements help no-one. What I hope, by raising this topic of conversation, is to facilitate small improvements to the requirements so that everyone has a better time of it, and software projects in general deliver better quality and better quantity.

Llyr Wyn Jones is a Senior Programmer at Grid-Tools, the leading Test Data Management Company.

He has invented, developed and implemented new software solutions for multiple worldwide clients. He tweets at @gridtools.

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

Quantitative Methodologies Assisting Performance Testing

- by Ramesh Viswanathan

Abstract:

The very important element in software development is Performance Testing, which is subsequently followed by Performance Engineering to ensure the application or the product developed are fine-tuned based on the outputs from the performance tests conducted and thereby ensure to meet the customer requirements and to append to this argument, always there is a need that life cycle of the application or product shall have higher availability and visibility. This paper starts with Introduction, the advantages associated with methodologies along with examples for each of the laws mentioned and finally with a sample e-commerce application that shall contribute in mapping the needs / requirements.

Introduction:

It is usually a misconception that performance testing activities under this testing arena is to basically use a load-testing tool to script the business scenario, execute the test and submit the results, but many are not aware the importance of basics (Quantitative Analysis / Methodologies) related to performance testing that are usually missed out or probably under the illusion that these are not required or might not come under their respective domain of work or nature of work activity. This gets very much misquoted when not properly communicated to the testing teams.

Equally important activity is, when test-leaders provide mentoring to their team-members and degree of importance that mentors need to provide / educate shall be of more value add at the initial stage which will for sure assist team-members at the later stage and provide a strong foundation before making their entry into performance testing arena.

Some say that these are related to statistics probably yes, but when the basics are understood and appreciated to the core they form the real essence and aid both during performance testing and engineering activities.

But what advantages do they provide is always that comes to every mind of the performance tester, though there might be few disadvantages but all these gets overridden by the positive factors. Let us understand some of the example and later part of the session understand the Quantitative Methodologies by going through few simple examples as to how well the quantitative methodologies assist at various stages be it requirement or analysis.

Advantages:

- Contribute in verifying and validating against the subjective statements defined during the requirements stage

- The requirements from the clients help in converting them into numbers that are essential during the scripting of the business scenarios
- Instead of blind usage of the numbers, they can be validated at the requirements stage and corrected to meet those numbers mapped to objective statements
- Quantitative methodologies benefit in providing the right kind of information for proper settings related to run time settings during test execution, let us take an example to understand this advantage, assuming that there are 100 users and in one hour they need to complete 500 transactions then the pacing that need to be applied between each iteration or loop for each user would be roughly around 12 minutes and by incorporating this value in the run-time settings of any load test tool the objective can be achieved
- The mathematical numbers provide more promising means of easy correlation between requirements, translation and validation
- The numbers speak more than words and when implemented both at the beginning, during and after test execution provides better validation
- These methodologies / quantitative approach is not only applicable to simple systems but provide foundations for complicated systems / applications

Some of the terms like Response Time, Utilization, Throughput, Queuing Time, Time to First Byte or Time to First Buffer, Service Time, Think Time and etc. can be correlated by means of Quantitative methodologies laws or rules as well.

There may be many rules or laws in this arena but some of the very frequently used amongst these Quantitative laws are as described below:

- Utilization Law
- Forced Flow Law
- Service Demand Law
- Little's Law
- Interactive Response Time Law

Let us define each of these laws with some examples and associate as to how they can be mapped to the requirements mentioned in the final e-commerce application;

Utilization Law: Let us consider a resource 'r', then the utilization of this resource is defined mathematically as the product of mean service time per completion in that resource **r 'Sr'** and throughput of that resource **r 'Xr'** and that is given by the equation **Ur = Sr × Xr**

Consider the example as to how we can implement this law, assume the bandwidth of a communication link is 1Mbps and it is used to transmit 20Kbyte packets that streams through the link at a rate of 5 packets / second. What is the utilization of the link?

Let us first understand what is given:

- Resource of the communication link is 'r'
- Bandwidth which is the maximum packets of the communication link = 1000000 bits per second
- The data transmitted by the link is 20 * 1000 byte ☐ 20 * 1000 * 8 bits / packet
- Throughput of the resource **Xr** is 5 packets / sec

Let us now calculate the service time **Sr** = data transmitted / bandwidth = 20 * 1000 * 8 / 1000 * 1000 = 0.16sec / packet.

Applying the utilization law gives **Ur = Sr × Xr --> 0.16 * 5 = 0.80 or 80%** as the utilization of the link.

Forced Flow Law: Let us consider the resource 'r', the relation between throughput of that resource (Xr), the average number of visits (Vr) made by a request to that resource and the throughput of the system (X0) is expressed by the formulae as **Xr = Vr × X0**

Consider a database system which is under constant load of transactions and assuming that all database transactions have similar resource demands and few metrics provided in the table below:

Disks	Reads Per Second	Writes Per Second	Total I/O s Per Second	Utilization
1	20	10	32	0.32
2	30	10	36	0.43
3	42	12	44	0.57

It is observed that in one hour almost 14,400 transactions are completed. Find the average number of visits for I/Os on each disk?

Mapping to what is given:

- The resource 'r' maps to each disk
- The observation period is one hour = 3600 sec
- The number transactions completed in that one hour = 14,400
- The throughput of each disk is given under the column Total I/Os per second
- The visits on each of the disk need to be calculated

Now, calculate the throughput of the database server **X0** = 14,400 / 3600 = 4 transaction per second

Therefore, referring the formulae **Xr = Vr × X0** calculate V1, V2 and V3

V1 = X1 / Xo = 32 / 4 = 8 visits to disk 1 per database transaction; V2 = X2 / Xo = 36 / 4 = 9 visits to disk 1 per database transaction; V3 = X3 / Xo = 44 / 4 = 11 visits to disk 1 per database transaction.

Service Demand Law: Let us consider the resource as 'r' and accordingly we have the product of mean service time per completion at resource r Sr and throughput of that resource r Xr is expressed by the formulae **Ur = DSr × Xr**

A Web server is monitored for 10 minutes and its CPU is observed to be busy 80% of the monitoring period. The Web server log reveals that around 36,000 requests are processed in that monitored period. What is the CPU service demand of requests to the Web server?

From this let us understand the given values:

- Resource is the CPU Service 'r'
- The monitored period is 10 minutes = 600 seconds
- The number of requests processed = 36,000 requests
- The CPU utilization is around 80% = .80

Need to calculate the CPU service demand?

The throughput of the webserver = 36,000 / 600 = 6 requests per second

Therefore the CPU service demand **DSr = Ur / Xr** = .80 / 6 = 0.133 sec per request

Little's Law: Let us consider the a system (closed system), the relation between the number of customers, the arrival rate of the customer or throughput of the system and the average time spent by the customer in the system is mapped by the formulae **Nr = Ar × Tr**

A computer system is monitored for one hour and around 10,800 transactions were executed and in this period of observation few jobs were executed, the average time spent by a job in the system is 10second, find the number of jobs that were executed?

Let us find the given details:

- Resource is computer system that is 'r'
- The monitoring period is 3600seconds
- The number of transactions executed were 10,800 transactions

Need to calculate the number of Jobs executed? Throughput of the system in the observed period = 10,800 / 3600 = 3 transactions per second

Therefore we have **Nr = Ar × Tr** = 3 * 10 = 30 Jobs

Interactive Response Time Law: This is similar to the Little's law but along with new variable called think time incorporated providing relationship between the average response time R, the number of clients which is either in the think state or waiting for a reply to a submitted request (M) and the system throughput (Xo) provided by **R = (M / Xo) - Z**

For a system that has processing speed of 3,600 requests during one hour by an interactive computer system with 50 clients and an average think time of 10 second, calculate the average response time?

Given details:

- Observation period as 1 hour = 3600sec
- Requests processed = 3,600 requests
- Number of clients = 50
- Think time = 5second

We need to calculate throughput as 3600 / 3600 = 1 request per second

Now applying all the values provided in the formulae **R = (M / Xo)** – Z we have (50 /1) – 10 = 40sec

The above laws provide the required basics to get started and shall form the foundation to get started with performance testing requirements understanding and also support during the performance engineering phase as well.

Now let us understand and e-commerce application and try to see how best these laws defined shall assist. Consider an e-commerce application; it was observed one business scenario for duration of 1 hour the number of transactions completed is around 28,000 which is almost the same number of transactions that arrived at the system. During this period the CPU of the system (webserver) was observed to be busy for almost 30 minutes. The response of system was set at 8 second for that business scenario with a think time of 15 second. Analyze the performance of the system?

Solution: When such type of application is to be analyzed or given understand initially to make a note of the measured quantities which is also known as operational variables and accordingly obtain the derived quantities which are based on the details mentioned in the performance testing laws; these are explained in the below sections.

The measured quantities also known as operational variables (defined / available) are as follows:

T → the observation period (1hour that is 3600 sec)
B_t → the busy time of the system observed in time T (30 minutes that is 1800 sec)
A_a → the number of arrival requests to the system in time T (28,000 requests)
A_s → the number of requests that was submitted to the system in the period T (28,000 requests)
C_c → the number of requests completions by the system in the observation period T (28,000 requests)
C_o → total number of requests completed by the system in the observation period T (28,000 requests)
R → response time as 8 sec and Z → think time as 15 sec

Now applying all these operational variables, let us understand the performance of the system;
Let us find the system throughput from the formulae given by $X_o = C_o / T \rightarrow 28{,}000 / 3600 = 7.78$ requests per second

A. Let us apply the Utilization Law to understand the utilization of the system, applying the operational variables to the formulae $U_s = B_t / T \rightarrow 1800 / 3600 = .50$ and therefore the **utilization is 50%**
B. Applying the Service Demand Law to understand the demand request, applying the operational variables to the formulae $D_{Sr} = U_s / X_o \rightarrow .50 / 7.78 =$ **0.064 sec per request**
But how do we validate is also a question that arises in every mind, here is the alternate way of checking this applying the mean service team per completion $S_c = B_t / C_c \rightarrow 1800 / 28000 =$ **0.064 sec per request**
C. Applying Interactive Response Time law to calculate the number of users, applying the operational variables to the formulae $R = (M / X_o) - Z$ ------------------------- (1), we need to find M;
Equation (1) becomes;

⇨ $M / X_o = R + Z$
⇨ $M = X_o (R + Z)$ thereby we have $7.78 (8 + 15) \rightarrow 7.78 * 23$
M = 178.94 approximately 180 users

Thus by applying the variables both measurable and derived variables the performance of the system is mathematically obtained that need to be validated by the load test tool during the test-scripting, test-execution and test-reporting phases.

Conclusions:

It is always better to understand the basics before getting into the related wider area of the subject; it forms a requirement or mandatory know-about moving forward about the quantitative methodologies in detail. Every activity within be it performance testing or performance engineering would need this strong skill of quantitative methodologies for better understanding of the application / product requirements for ensuring quality output of the application / product that is tested.

References:

- Computer Capacity Planning: Quantifying Performance Models By Virgilio A.F. Almeida, Lawrence W. Dowdy, Daniel A. Menasce May 13, 2004
- Capacity Planning for Web Services: Metrics, Models, and Methods by Daniel A. Menasce, Virgilio A. F. Almeida, Prentice Hall, 2002
- Capacity planning and performance tuning from Oracle documentation site http://docs.oracle.com/cd/E13214_01/wli/docs102/capplanguide/process.html

Ramesh Viswanathan is Master of Engineering in Communications Systems and a Master of Business Administration in Operations Management, and is both Siebel and ISTQB Certified. He has been in the Software Testing industry since 2001 and worked with various organizations like ReadyTestGo, Cognizant Technology Solutions, Symphony Software Services, ANZ Bank Information and Technology, SunGard Global Solutions and presently working for a US based MNC as Sr. Performance Test Architect.

Road to Test Automation

- Part 3

- by Georgios Kogketsof

Page Objects Are Not Enough

In my previous articles I presented an abstraction-layering model for the automation testing implementation, to sustain maintainability. In the aforementioned model a layer was dedicated to page object pattern that is one of the most popular patterns in test automation just because it promotes test maintenance and unifies the way QA teams work.

As the pattern dictates, a page object is the representation of an html page using objects. A page object contains the html element addresses (Xpath, Css or DOM) and the user actions on them as methods (press, input, select, etc.). Although the page objects pattern is widely accepted I cautioned the readers that page objects were not enough and now I am obligated to elaborate on that.

My work experience showed that the pattern is inadequate to fully describe an html page because it lacks the ability to represent the business logic of the page. In order to solve this problem I started using a complementary entity to the page object, the business object as shown in Figure 1.

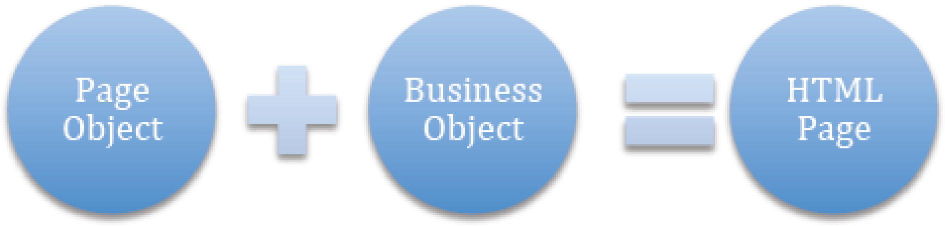
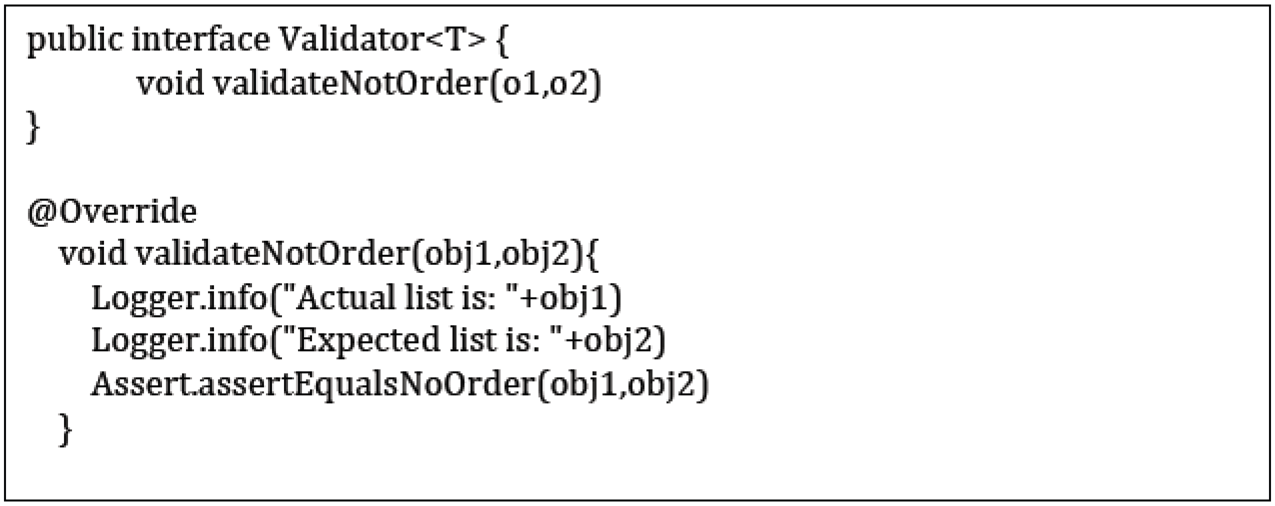


Figure 1. Html page representation

The business object is a class containing the business logic behind the page. An example of html page business logic is a form's mandatory field validations. The page object as an entity does not contain any logic, it only describes user actions for example pressing a "submit" button, if an error is raised as a consequence of the action its agnostic of it. The business object complements the page object in having a method for describing "submission and error raised". The complementing nature of the business object stands because the business objects uses the page object's methods to assemble the logic thus its methods are a collection of page object methods.

The breakthrough of this approach in the representation of the html page is that it keeps a clear separation of the logic from the html elements introducing an abstraction layer for the business object thus separating future changes in the page logic from its locators. Using the previous example a possible subtraction of a mandatory field does not affect the page object only the business one thus the change is marked in a single place.

Business objects should contain compound commands and assertions. Compound commands are considered a group of user actions each one contained at a page object. An example of a compound command is the update of a field input resulting from logging to the application, navigating to the desired page and updating the value of the field. In this case the business object should contain pages objects for the login, the navigation and the update of the field. The result of this series of action should be an assertion. The assertions found in my business objects are always in the form of validators. Validators offer the flexibility to switch between assertion offering frameworks such as Junit, TestNG and SpringSource. A very simple example of a validator is demonstrated below in Figure 2:



```
public interface Validator<T> {
        void validateNotOrder(o1,o2)
}

@Override
   void validateNotOrder(obj1,obj2){
      Logger.info("Actual list is: "+obj1)
      Logger.info("Expected list is: "+obj2)
      Assert.assertEqualsNoOrder(obj1,obj2)
   }
```

Figure 2. Validator Example

The validator of the above example is used to compare two lists independent of ordering.

The relationship between a page object and a business object could be one to many and let me explain what I mean. In page object theory when we have more than one page of the same content we can implement one global page object or create page objects for each repeating page. I personally prefer the second solution because the danger behind declaring only one page object is that for all the identical html pages using their locators outside their page objects violates the fundamental principles of PO (locators belong in one, and only one Page Object). In this complicated schema I prefer not complicate things more thus I use only one business object for all identical page objects as shown in Figure 3:

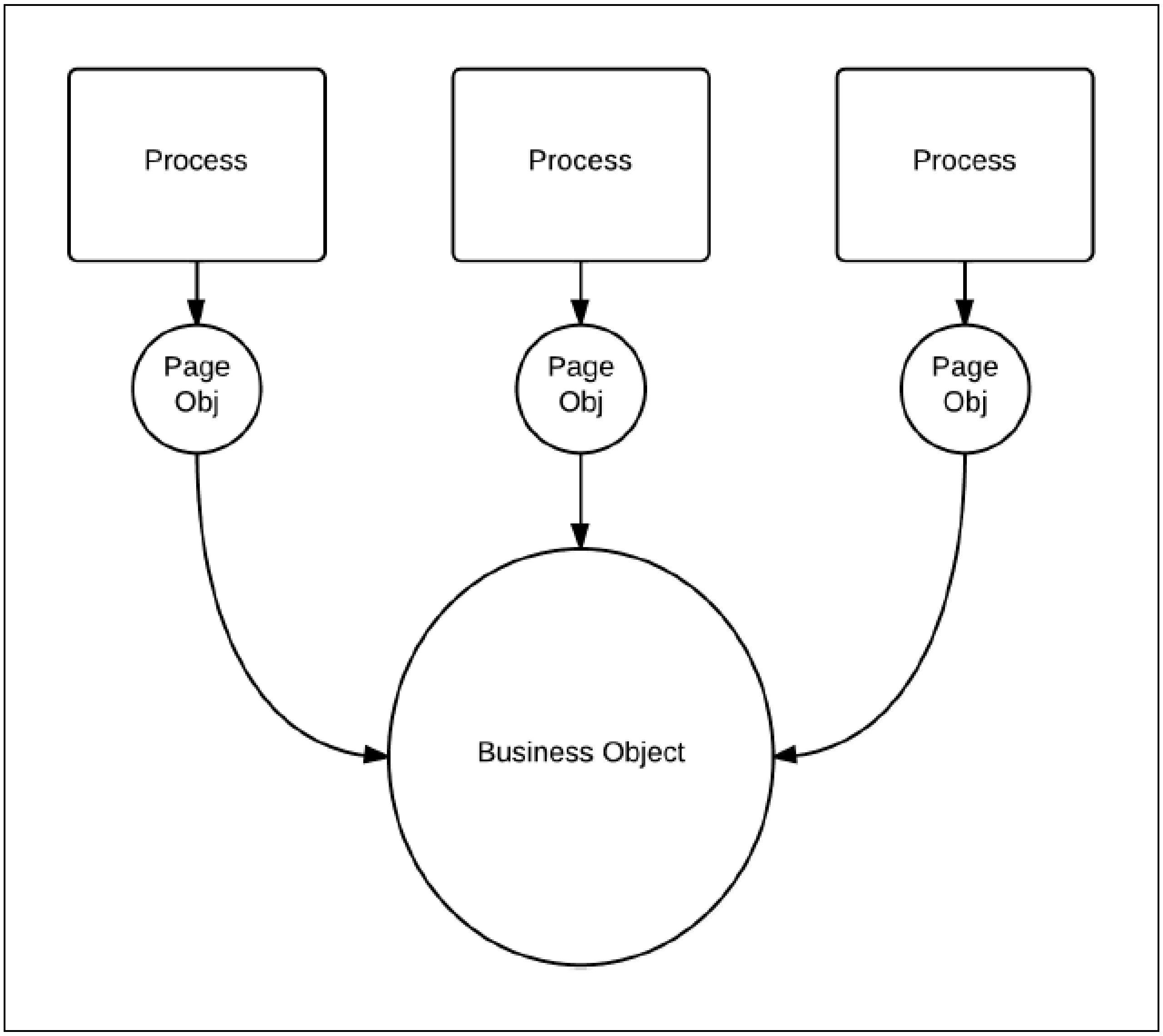


Figure 3. Business to page objects relationship

This newly created entity changes our proposed abstraction layering by introducing a new layer under the page objects the business objects layer as shown in Figure 4 on next page.

One more difference from our previous model is that now the common functions vertical layer does not support the page objects anymore and that make sense now that we have striped the logic out of them.

At this point I should mention that I do not claim the parenthood of the business objects. The need for their creation came out of work experience and I could not find any relative information when I searched for a solution to my problem. The business objects may exist and can be referred differently.

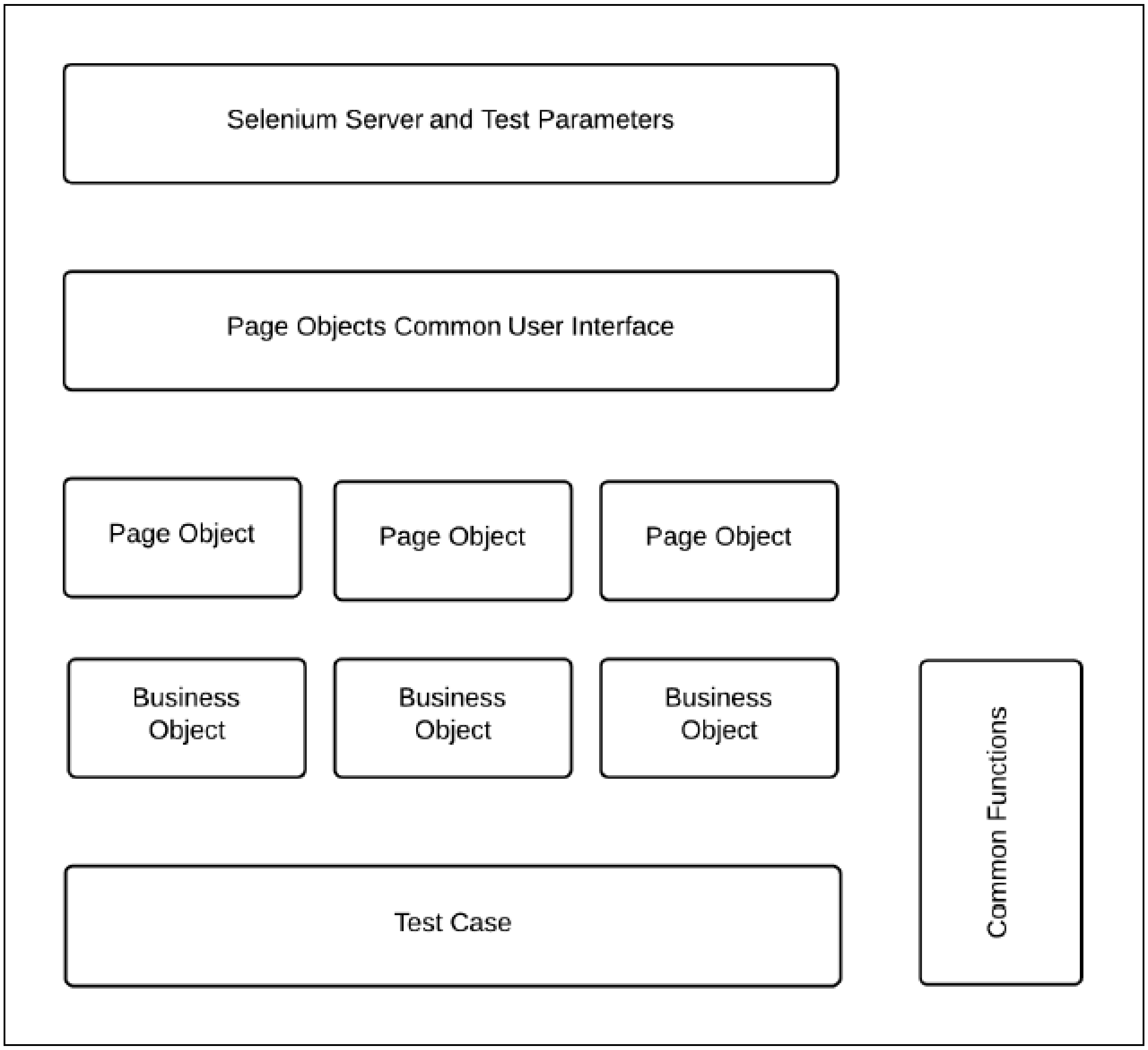


Figure 4. Abstraction Layering Model

To be continued in next issue…

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

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

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

Sharing is caring! Don't be selfish ☺

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

CONTINUED FROM PAGE 13...

Yes this is not something easy to do but it is an important job you have when recruiting and you better take it serious. It is time for you to raise the bar!

What are the skills you actively seek out while hiring new testers?

Curiosity, guts, and willingness to learn are three key skills I look for. I look at personality, preferences and ethics when I hire co-workers to House of Test. I don't highly value specific technical or domain knowledge. It is great to have it but if you have the skills I mentioned, you won't have any problem acquiring the technical skills needed when facing a challenge. I want to know what drives you. I believe that to become great at something, energy and motivation must come from inside yourself and not after getting pushed from outside. I don't really care what it is you want to be great at as long as it is something. I only hire people who want to be among the best at something and also have the confidence that they can reach that level.

You mention that you fight the template zombies and certification quacks. Why is that?

Because they are IMO not doing any good to our craft. They are preaching that testers are interchangeable and that the greatest value are in documents. This is an over simplified view on testing developed 30 years ago or so and today it is completely outdated. But it is an easy sell to management which provides a fake impression that valuable testing is done by heavy upfront planning, is controlled by metrics and that everyone is doing the same thing.

It is time to acknowledge the complexity of testing and the skills that are needed to be a great tester. Here is an example for you: go to a three day ISTQB training ending with a multiple choice questioned test and get 60% of them correct; you'll receive a certification (that you now have foundation level knowledge about testing). Compare this to what we are doing at House of Test where we run a 1.5 year full time professional education in software testing.

This training is run by internationally recognized testers with the focus on developing skilled and thinking entry level testers that see the diversity in testing and know that there is no *one size fits all* solution. These students are studying and learning through experiential exercises with tons debriefs and of critical thinking. After 1.5 years, my belief is that we do reach to what I would call a foundation level of skills needed to become a great tester in future. I hope, you as a reader clearly see the difference of what ISTBQ thinks is required to call someone an entry level tester and what I require before I can vouch for student to possess entry level skills.

To be a valuable tester requires hard work and serious practice, there is no shortcut called certification. Unfortunately there is a whole horde of people who have taken the certification only to be fooled by ISTQB to think that they are great testers.

If you hear someone talk about their company developed and branded best practice that promise to solve your testing problems independent of your context; you have good reasons to suspect you are running into a quack.

What impact Context Driven Testing has made on your testing career?

I think the greatest impact it had on me is that I'm still in the testing profession. The CDT community is truly a wonderful bunch of people and many of them have become very close friends to me. So in the dark hours when you are over flooded by stupid ISO testing standards or commodity testers, you start to lose hope in the future of testing. This is when the bitterness creeps up on you and you start thinking that it is time to leave testing and to do something else. This is when I turn to my friends and positivity in my community to get reminded that there are great things going on. And even though we are still small in number, we are gaining a ground. And it is CDT community that is coming up with the new ideas on taking our craft forward.

Testing books and blogs that you read and follow regularly are….

Pick up any book written by Gerald M Weinberg and you'll have an interesting journey ahead of you. I also recommend looking into the work Dan Ariely has done on behavioral economics, which I think relates to testing very well.

We are curious to know about your involvement with ISST.

I'm one of the four founders of ISST (International Society for Software Testing). Over the years in many cases, testing has become a simplified commodity that does not bring much value to anyone more than consultancies selling services. ISST's mission is to put common sense back into testing. There are too many off the shelf test methodologies and processes that promise high quality testing; completely independent of context and problems that need to be solved. The new testing standard ISO 21991 is a great example of this. We oppose practices that are wasteful or that seek to dehumanize testing! We advocate an approach to software testing that emphasizes value and the role that skilled testers play in its delivery. Within ISST, we have a couple of initiatives going on such as education, advocacy and of course the Stop 21991 movement. Check out commonsensetesting.org and if you support our cause, don't hesitate to join us. And do your part for a better testing world!

And how was your experience with CAST 2014 conference?

I have been to seven CAST (all but 2007 & 2013) and this year was hands down the best CAST I've been part of. A very well arranged conference, the venue was nice and it was a positive and energetic vibe at the conference. I think that CAST crew really nailed it this time and also having it in New York was a smart thing to attract us from Europe. Maybe this was one of the reasons why it was the first time they sold out.

If you are given a magic wand to change one thing about testing field, what would that one thing be?

Sorry! I don't believe in magic. Instead I try to do something that moves us forward and I encourage every reader to start doing something. Don't sit around and hope for some magic to happen or longing for someone else to do something. It is time to step your game up!

What are your future plans?

There is one brand new thing that I'm involved with that gets me going. Looking around conferences, I see a huge chunk of the same speakers that travel from one conference to another. It is time for new blood to get more diversity into the conferences. When we had call for proposal for Let´s Test, we specifically reached out and asked for new speakers but even after doing so, we did not get much new people wanting to speak at Let´s Test. I don't think there is a lack of really cool experiences out there but I think that for many people it is a huge step to jump up on a stage and share their experiences. Anne-Marie Charrett and Fiona Charles have started a very important initiative to mentor and support new speakers. I got the honor to be a patron for this important cause. I'm very thrilled to see all the new speakers coming out of this and spreading out all over conferences over the world. This is very much in line with what I care about, for building our community. Have a look at speakeas.ie and if you like what you see, please don't hesitate to be part of it. Either as a new speaker or a mentor, volunteer, sponsor.

The future for House of Test is looking bright and shiny. CDT is gaining ground and "no-bullshit" companies like us are getting more and more respect and attention. During 2014, we grew by 60% and we will continue our expansion during 2015. But rest is assured that we will never lower down the bar to become a Hottie. We are still only inviting Crème de la Crème to join our house.

Let´s Test is still rocking hard. In 2014, we had conferences in Sweden, Netherlands, Australia and South Africa. For 2015 we have announced conferences in Sweden and the Netherlands but don't worry. We will pop up on other locations during the year, just wait for it……

I have thoroughly enjoyed our discussion so far. Any message to our readers?

Get your ass of the couch. You are smart and have interesting experiences that are important to share. Being part of our lovely community will reward you with advancing your skills in testing, deeper insights into testing, vivid discussions, critical thinking, gaining reputation, life time friendships, lots of fun and plenty of beer.

Last question. What is your opinion about Tea-time with Testers? We would love to get your feedback and suggestions.

I love what you are doing! You are making it possible for testers to read what is happening in the world and to get to know other testers a bit more. By this, you are helping in growing our community and providing access to readers who can't travel the world to be at conferences or workshops to pick up the latest thoughts through your magazine.

Thanks you Henke. It was great talking testing with you! Our best wishes for your future journey.

Happiness is....

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

Like our FACEBOOK page for more of such happiness

https://www.facebook.com/TtimewidTesters

Download Free NOW!

STM

NOVEMBER - 2014

YEAR I ~ ISSUE I

STATE *of* TESTING

Know more. Before it's too late!

Live long and Prosper

Expertise creates the path with how & what-to-do whilst
Experience smoothens it with what & when-not-to-do

It is that time of the year when we reflect on the year gone by and look forward to another exciting year. Another year added to our work experience and hopefully new expertise acquired. And we look forward to the New Year to deepening our expertise.

Experience and Expertise - Mulling over

"Experience" is knowledge or skill that is acquired over a long duration of time (years normally) whereas "Expertise" is knowledge or skill that is acquired irrelevant of the number of years, but rather from meticulous training and rigorous practice.

Experience is gained by solving real problems, irrespective of the outcome being successful or not. An unsuccessful outcome teaches one 'what-not-to-do' which is very useful in future situations, while a successful outcome sharpens the skill and boosts confidence.

Expertise is about deep knowledge in the given area. It is knowing in depth about techniques, principles and guidelines that enable problem decomposition and solution formulation. Expertise is built by learning, practice and reflection, trying new things and challenging yourself and finally watching other people in action.

Is experience sufficient? "Live long"

An experienced person may be able successfully solve the problem, but may have difficulty in explaining as to why it is the best solution. And this is where expertise comes in handy - the technical ability to decompose the problem, formulate various solutions, pick the optimal one and justify this logically.

Doing an activity multiple times over long duration does build experience, but this has the risk of dulling the senses so much so that it can become a rote custom. And that is when it becomes useless/dangerous. Adapting to the current situation, extracting and refining heuristics is what makes experience really valuable. Living long requires continual adaptation.

Is expertise good enough? "…and Prosper"

Deep knowledge is useful only when you know how to apply this. Rigorous practice converts the knowledge to skill, and this is useful to solve problems. Expertise enables us to "what and how to do", but may not be also allow to ascertain "when/how not-do-". The latter is normally derived from experience. "Prospering" requires depth of subject matter knowledge/skill without which life would be shallow.

You can transfer experience, not expertise

Unlike expertise, experience isn't anchored to a specific context. Rather, it confers generalized competencies that be transferred to a different setting. The judgment and maturity accumulated by working in a variety of roles and settings can be called upon in different situations. Experience is proven by past behaviour, not present knowledge.

Experience as such does not matter. It is what you do with it, which counts. Can you apply what you have learnt to new situations?

What does it take to build expertise?

A strong pre-requisite to building expertise is strong interest in the subject matter. Without deep interest or passion, it is very difficult to be devour the knowledge via training or do repeated practice. Expertise can also be gained by experience, by doing repeated work, ideally solving real life problems. However, not always does experience gain expertise. Many people work in a particular field and still never manage to learn anything new about it!

Expertise does not come easily - It requires one understand the fundamentals, to think deeply, and challenge yourself and step outside what you have experienced.

Hire based on experience or expertise?

Interesting question eh… Let us examine two cases…

Case #1:

I have often seen customers focusing on years of experience whilst they actually want deep expertise. In a recent customer engagement, one of our customers had scaled their test team rapidly with experienced lateral hires and they had tremendous issues. The reason - the lateral hires had calendar experience, but did not possess a strong testing expertise.
The result: The team had difficulty in understanding the (new) product, resulting in ineffective test strategy and test cases. The lack of testing expertise despite years of testing resulted in shallow knowledge that could not be applied effectively in the new context. And hence a large number of customer escapes.

Case #2:

A few years ago we worked with a customer whose head of engineering had an ambitious automation goal. And the test team with deep expertise on automation had done a good job of creating a large script base. But the requirement was to support a multi-platform distributed application, which the team found it difficult to handle and therefore concluded that the tool had limitations and hence the ambitious goal could not be achieved. Our experience enabled us to understand what-not-do-do and therefore we solved the problem differently by extending the existing tool set with another technology wrapper.
How does this matter when you shift careers?

It is important to understand between distinction between expertise and experience especially when you are shifting career. The act of shift can be accomplished via acquiring new expertise or using your experience in a new setting ("transferable abilities"). The latter is easier when you have reached mid-career. Note that both do present considerable risk, a major career shift should never be undertaken lightly.

So what do you want to accomplish in this year?

Another wonderful year has started and you must have made plans for this year. Think in terms of new expertise you plan to acquire and become deep, be it in the areas of testing, technology, tooling, process, management. What are the new techniques, principles that will enhance the knowledge? What activities are you planning to perform to enhance your experience? And how do you plan to learn from the experiences, formulate heuristics and become not only older but wiser? To another year of rich experience to broaden your horizons and deepening knowledge/skill and enjoying the journey...

I believe strongly in deep expertise to build a higher potential, a positive can-do attitude to leverage this and respect good experience that enables constant unlearning to facilitate new learning to apply in new contexts. Hey, enjoy the journey!

Cheers! Let us toast to your growth. Wish you a wonderful 2015.

As the logical Vulcans in Star Trek say, "Live long and prosper".

Back To Index

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

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

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

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

Advertise with us

Connect with the audience that MATTER!

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

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

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

Well, hear directly from them.

And the **Good News** is…

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

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

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

in nexxt issue

articles by -

Jerry Weinberg

T Ashok

Joel Montvelisky

Georgios Kogketsof

...and others

our family

Founder & Editor:

Lalitkumar Bhamare (Pune, India)

Pratikkumar Patel (Mumbai, India)

Lalitkumar

Pratikkumar

Contribution and Guidance:

Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)

Jerry

T Ashok

Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover page image – theresakistel.com

Core Team:

Dr.Meeta Prakash (Bangalore, India)

Unmesh Gundecha (Pune,India)

Dr. Meeta Prakash

Unmesh Gundecha

Online Collaboration:

Shweta Daiv (Pune, India)

Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)

Kiran Kumar

Chris

Romil

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

To get a **FREE** copy,

Subscribe to our group at

Google

Join our community on

facebook

Follow us on - @TtimewidTesters

www.teatimewithtesters.com

Give Feedback