

THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

APRIL-MAY 2013 | YEAR 3 ISSUE II

Jerry Weinberg

Direct Observation of Quality

Ben Kelly

The Testing Dead – part 3

Anne-Marie Charrett

Where are All Good Jobs?

Leah Stockley

Addressing the Risk of Exploratory Testing -2

Johanna Rothman

Interviewing to Detect Great Testers

T Ashok

The Onion Peel and Half-filled Tea Cup

Joel Montvelisky

Coordinating your automated and manual test management



The Best of Tea Time

Coming soon...



TEA-TIME WITH TESTERS

First Indian Testing Magazine to reach 101 Countries in the world !

Created and Published by:

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

Editorial and Advertising Enquiries:

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

This ezine is edited, designed and published by
Tea-time with Testers.

No part of this magazine may be reproduced,
transmitted, distributed or copied without prior written
permission of original authors of respective articles.

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

Editorial

One more reason to smile



They say that it helps when testers and programmers understand each other well. And for that it is important to understand each other's language too.

Remember Lisa Crispin's wonderful article '[Learning to communicate better with programmers](#)'?

Well, I have good news for you. We have partnered with 'Software Developer's Journal', a popular magazine in community of software programmers. They are offering special discount to TTwT readers, so that you can update yourself with latest in programming field and can communicate even better with your programmer friends.

Actually, some more good things are in store but that require more commitment from my end and hence I am offering you another mega issue for April and May.

Worry not. We haven't compromised on content 😊.

Enjoy reading. See you next time!

Yours Sincerely,

A handwritten signature in purple ink that reads "Bhamare".

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



QuickLook



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

The Testing Dead (part 3) - 16

Where are All Good Jobs? - 19

In the School of Testing

Addressing the Risk of Exploratory Testing-25

Interviewing to Detect Great Testers - 30

Coordinating your Automated and Manual test Management - 38

T' Talks

The Onion Peel and the half-filled Tea cup- 43

Testing Puzzle – S.T.O.M. Contest

Family de Tea-time with Testers

Testing Puzzles
by Sebi



Are Software Testers Involved in High - end & Challenging Work?

By – SiliconIndia

Software Testing Professionals are not involved in high – end and challenging work as mentioned in talentspirit.com.

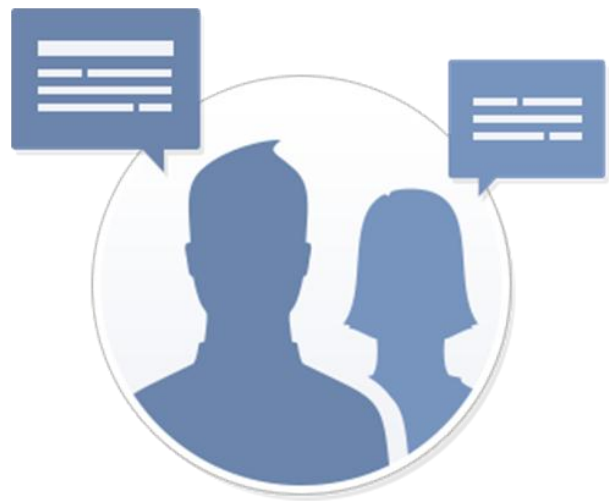
According to Ramana LRV, Head of Software Testing Practice, TalentSprint, "With the lines between development and testing blurring, a test engineer is no longer considered as someone good at clicking a button on the application's user interface. There are many more activities and skills expected from a test engineer."

When the activity of a software tester becomes more practical, they can be more effective and beneficial to the organization. This is the reason why many term software tester as professional investigators. One of the biggest responsibilities of a test engineer is to ensure that the product meets the quality standards. They should always remember that the success of the product highly depends on the quality. Thus, a tester should not only have excellent technical skills, but they should also have good communication skills, analytical and judgment skills as well as leadership qualities. Apart from the mentioned skills, a software tester should be able to think out of the box, be critical and creative and always aim for excellence. When communicating, they should be able confidently converse to various groups of people. They should also have some passion towards software testing.

Having only technical skills and knowledge isn't enough for software tester. However, they should also have a good domain knowledge which includes their ability to write codes, understand the architecture of the application and the system.

Software testers are expected to be familiar with most of testing tools as they will one day need to use them. They are also expected to be familiar with the various software testing terms such as Agile testing, Risk based testing and others. If you are planning to specialize in a specific domain such as banking or insurance, then it will be better if you get certified in the particular domain.

**You are
invited to
discuss this
on our...**



Software Developer's JOURNAL

new ideas & solutions for professional programmers

Check out our new issue – MongoDB!



We're very happy to announce that our issue about MongoDB is now available!

You can download a free article by clicking on that link:

<http://sdjournal.org/>

We've also prepared a special 20% discount for TTWT readers! Enter this code while subscribing: 20ALSDJ

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Tea & Testing



with

Jerry Weinberg

Direct Observation of Quality (Part 1)

"I think there is a world market for about five computers." - Thomas J. Watson, Sr. founder and Chairman of the Board of IBM, 1943

"There is no reason for any individual to have a computer in their home." - Ken Olson, founder and President of DEC, 1977

Although, software is different from other things we create, it's not totally different. For one thing, it invites speculation about its value, just as computers have done. For another, as the above quotations show, when it comes to speculation about future value, one expert's guess is just about as good as any others. Here's a couple of good examples of software predictions, which in all modesty I should add to the quotes above:

"FORTRAN will never catch on." - Gerald M. Weinberg, 1956

"FORTRAN will never last." - Gerald M. Weinberg, 1966

In short, speculation is not the way to produce quality software. To do that, you need to know many of the same things you'd need to know for any quality product—pearls or puppies, bicycles or books—based on fact, not opinion.

And one of the things we need to know—if we believe the cybernetic model—is this:
What is the quality of this product, right now?

There are two different approaches to answering this question: the direct approach and the indirect approach. The direct approach measures the quality. The indirect approach measures something else and then adds meaning to the measurement in terms of quality. Thus, for instance, my word processor can compute the Flesch Readability Index of my book from the average sentence length and word length. If this number is between 10 and 11 (which indicates a high school sophomore or junior level), I can take that to mean that it is a good quality book. Some pitfalls of this indirect approach are evident:

- There may be more to quality than readability (such as content).
- There may be better measures of readability than the Flesch Index.
- An index of 9, or 12, might mean better quality than 10 or 11.

Given all these pitfalls of indirect measurement, it's a good idea to take a stab at direct measurement before going around the long way of indirect measurement. Direct is better. In this chapter, we'll explore direct measurement.

Quality versus Apple Pie

One of the things I've learned about creating quality books is it takes a lot of work—and that's certainly true of quality software as well. To avoid unnecessary work, I try to determine if there's a market before choosing a topic for a new book.

A survey of attitudes about quality

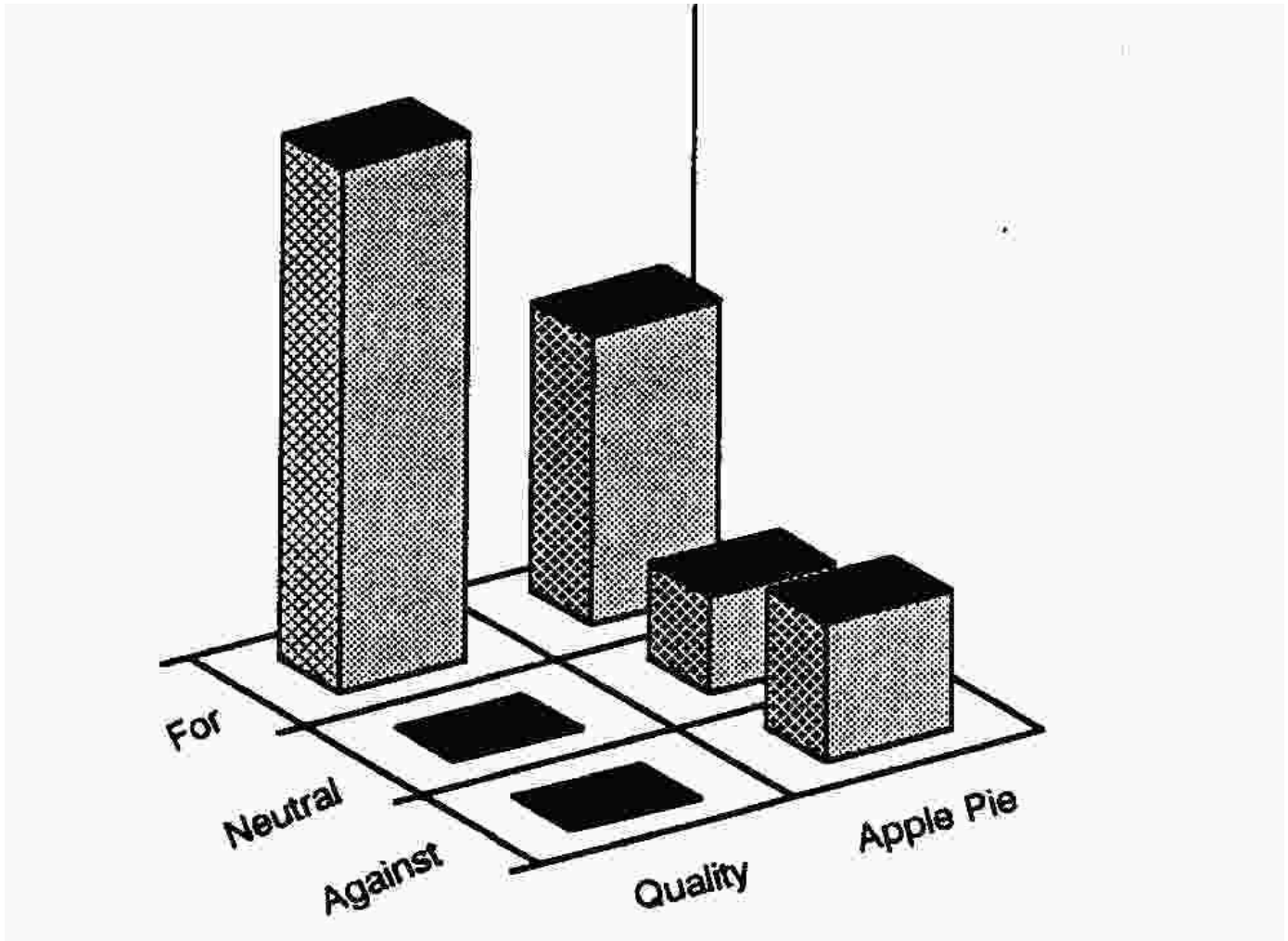
Before starting to write a book on quality, I surveyed 38 people on their feelings about quality. I asked each of them, how do you feel about quality?

- I'm for quality.
- I'm neither for nor against quality.
- I'm against quality.

In order to be able to place meaning on my data, I needed a control. I asked each of them, How do you feel about apple pie?

- I'm for apple pie.
- I'm neither for nor against apple pie.
- I'm against apple pie.

Figure 7-1 shows the results of my survey. Clearly, some people didn't care much for apple pie, but nobody said, "Actually, I don't much care for quality." It looked like the book was a good idea.



(Figure 7-1. How people feel about apple pie and quality.)

What do people mean by quality?

To determine what this survey meant, I used the steps for interpreting observations. I formed some hypotheses about the meaning of Figure 7-1:

1. This was a good year for quality.
2. This was a bad year for apples.
3. "Quality" is such an ambiguous term that nobody could be against it.

I chose the third hypothesis and set about testing it.

Generally, we cannot interpret a single measurement in isolation, which is why I used "apple pie" as a control. I also did something else to check this hypothesis. I asked each respondent to write, in 25 words or less: What does "quality" mean?

Here, the results were much less clear. From the 38 respondents, I got 31 different definitions of quality, and 7 who said that quality couldn't be defined! (Most of them could define "apple pie" pretty well.) I concluded that my survey question about quality was equivalent to:

How do you feel about what you like?

- a. I'm for what I like.
- b. I'm neither for nor against what I like.
- c. I'm against what I like.

In other words, the survey means that everyone has the same definition of quality, and it goes something like this: Quality is whatever I like.

What does quality mean?

With this underlying definition of quality—whatever I like—I found it curious that none of the original definitions mentioned people at all. It was as if "quality" was something that existed in nature, independent of human activities, opinions, or intentions.

More precisely, I should say that none of these definitions mentioned people explicitly. If you read them more carefully, however, you'd see that there were always people there, like the Wizard of Oz, "behind the curtain."

For instance, one definition said, "A thing that has quality is the best of its kind." To me, that definition implies a ranking process, and that implies someone to do the ranking. Even if the ranking is done by a machine, then someone had to program a ranking algorithm into the machine.

Another definition said, "Quality is the indispensable element." "Indispensable" implies a person or persons who cannot dispense with that element.

A third definition said, "Quality means having plenty of good properties." But "good" doesn't exist in the universe unless there's someone to make the judgment.

I've done this apple pie experiment many times, and each time the group fights tooth and nail against my detection of the human factor in their definitions. Often they will challenge me with quotes from experts on quality, like Philip B.

Crosby, who says that "Quality is conformance to requirements." Then I ask where those requirements came from— who "requires" them?—and the wrangle resumes.

Quality is relative

When I teach about quality, I always provoke this people/quality fight early in the class. Why am I so disputatious? Because if we don't get the issue out in the air, it festers like an abscess throughout the class, preventing all progress on the subject of quality.

Eventually, I get them to see the reality of my definition:

The political/emotional dimension of quality is made evident by a somewhat different definition of quality. The idea of "requirements" is a bit too innocent to be useful in this early stage, because it says nothing about whose requirements count the most. A more workable definition would be this:

"Quality is value to some person."

By "value," I mean, "What are people willing to pay (do) to have their requirements met."

Why is this definition so troublesome to some people? There are people in the world whose strongest desire is to find perfection—the one right way. Many of these people choose to work with computers, and particularly to go into the software business. They don't like the idea that quality is relative—to people, to place, to time. But it is.

Perfectionists usually claim to be devoted to truth. In that case, they should recall Alexander Pope's observation: "One truth is clear: Whatever is, is right."

Quality is relative, so the relative definition is right.

to be continued in next issue...

[Back To Index](#)



Biography

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



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

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

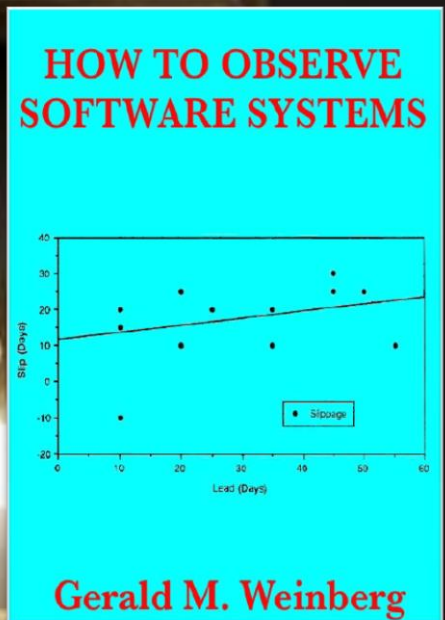
To know more about Gerald and his work, please visit his Official Website [here](#).

Gerald can be reached at hardpretzel@earthlink.net or on twitter [@JerryWeinberg](#)

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

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

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



TTWT Rating: ★★★★★

A photograph of a green, conical pendulum bob hanging from a thin wire. The bob is positioned over a surface of light-colored sand. In the sand, there is a large, circular, swirling pattern that resembles a fingerprint or a stylized mandala. The text "Speaking Tester's Mind" is overlaid on the image in a large, blue, serif font with a white outline.

Speaking Tester's Mind

- straight from the author's desk



The Testing Dead

episode-3

Surviving a zombie attack

- A series by Ben Kelly

[Please read part 1 and 2 of this series in previous issues of Tea-time with Testers]

Previously on the Testing Dead I talked about various forms of behavioral dysfunction that I call **Zombie Testing** and **why that's a problem**. So what can you do if you find yourself with a zombie infestation at your place of work?

Well you could fire them,



but you may not want to make that your first action. Some zombies can be rehabilitated.

Some are thinking human beings mimicking zombie behavior because either they don't know any better or perhaps they don't feel it's their place to break the mold. So before you consign your zombie testers to the sweet flames of napalm death – do have a look around for ones that show signs of higher brain function and help them if you can.

Pairing them with experienced testers is one way of helping them learn new skills. Some testers with zombie tendencies believe they are skilled testers because they don't know what they don't know. Pairing them up with someone who **is** a skilled tester can help impart skills that they didn't know they needed.

If you can, send them to courses such as **Rapid Software Testing** - if that doesn't light a fire under them, nothing will.

For the ones that don't want help becoming a better tester – You cannot force change. All of the helpful links to blog posts and exercises and offers of coaching will not avail someone who isn't interested. If that is the case, perhaps you can help them out the door instead.

Identify how zombie testers get into the building and apply liberal defensive countermeasures. Take an active hand in recruiting of new testers. You may need to help educate your HR people or recruiters to get past buzzword bingo and be able to identify real testers, or at least be able to not throw away promising candidates because they don't have the currently most fashionable acronyms in their resume. I've worked with a score of recruiters over the years and perhaps one or two had any real clue about what testing was. The typical testers resume I get from recruiters generally looks like this:



Most recruiters I've worked with were open to learning more about testing. If they're good, they'll want repeat business from you, so it's in their best interests to find out what sort of people you want. This is

not an overnight process. You're not going to sit someone down and lecture them on all the stuff they need to know. Tell them what you want up-front, sure. Give them the details that are important, but I find that real understanding takes more time. Develop a relationship with recruiters that make an effort to provide you what you want. Catch up for lunch occasionally or a Friday afternoon beverage of choice and talk about what's going on in the industry. Talk to them about the frustrations you have when hiring. Doing this has been worth the time investment for me.

Give them a profile of the sort of person you're looking for. If you're looking for someone more experienced, then someone with a diverse background and a number of different skills – there's a marked difference between someone who has 12 years experience and someone who has 3 years experience 4 times. Testers who are constantly honing their skills. They should be able to talk about what they do to stay sharp. I wrote a post a while back about what I look for in a resume. I share that sort of information with recruiters also. Make sure they know that not having certification is okay.

On that subject, I think certification sometimes unfairly demonized. If people get value from studying for a certificate, that's fine. I don't think certification creates zombie testers, although I do think it acts as a neat form of spray-on credibility for them. My major objection to certification in its current form is that it is marketed as a measure of tester competence. It is no such thing. I would love to see certification bodies be more up-front about this, but they have a financial disincentive to do so – they grow fat off the ignorance of testers and the people that hire them alike. Quite brilliant in a morally bankrupt kind of way, but I digress.

Rehabilitate the zombies you can, get rid of the ones you can't and do what you can do to keep any more from getting into the building you work in. What about the wider testing community? The truth is, with the barrier to entry to the testing profession being no more difficult than knowing where a computer's on switch is, zombie testers are being churned out far faster than we can hope to rehabilitate them. Does that mean it's a lost cause? Far from it I think, but focusing on the zombies won't get the job done. I'll talk about that some more in the fourth (and final) article of this series.

Ben Kelly is a software tester living and working in Tokyo, Japan.

He has done stints in various industries including Internet statistics , insurance and most recently online language learning.

When he's not agitating lively discussion on other people's blogs, he writes sporadically at testjutsu.com and is available on twitter @benjaminkelly.





[Sketchnote by @rosiesherry]

My personal experience has been that yes, on one or two occasions I've failed to be interviewed based on lack of certification. Rather than see this as a negative; I see this as a blessing. After all, if your idea of a tester is that narrow, then I'm probably not suited to your company. On occasion I've cited 'NO CERT!' to justify why I can't apply for a role. "I'm so sorry, I'm not ISTQB certified..." as I edge my way to the door.

I think it's naive to rely on a certification as a means to getting work. Especially if you are a thoughtful and intelligent tester who cares about the quality of your work, and who wants to be taken seriously in the industry.



But forget about certification for one minute. Are you seriously willingly going to put your career into a stranger's hand who then decides your fate by a keyword?

There are cleverer ways to play this game.

Have you not noticed that the way companies recruit is rapidly changing? To get a job that interests you, it's not good enough to send in your resume or hold a silly piece of paper with an over ornate stamp on it.

Those days are long gone. Now you need passion, you need to keep up to date with what's happening in your field, you need to be committed to keeping yourself relevant.

Our family has experienced this first hand when my husband was looking for work last year. He suddenly discovered at the 'old age' of 42 that he was unemployable. He is intelligent and personable (yes I am biased) with a first class degree in Electrical Engineering and he had made the assumption that good people always find work. But that's not enough. Today, if you want a job that's worthy of you, you need make sure you earn the companies respect.

This is not only based on my personal experience, I've spoken to many recruiters in the last few months, and all seem to have similar stories. Companies are more reluctant to use recruitment agencies to find their staff. They want recruiters who know and understand the specific skills they are seeking. I'm seeing recruiters leave the industry, or re-invent themselves as specialists in one field. Other ways the industry has changed is that many recruiting companies work for one company and are in effect the procurement arm of the company.

Of course, the testing industry has changed significantly too. Now that the major consultancies have successfully sold testing as a commodity, testing (or an excuse for testing) is being performed wherever people are cheapest. The adoption of Agile as a development process and its dependency on automation has also reduced the need for testers (though this is not necessarily a bad thing). The fact is, there are less testing jobs out there.

That doesn't mean there are less quality testing jobs though. While it's true that the crumbs from the table need to be shared among more, there is still plenty of meat and gravy at the table. The question is, have you earned a spot there? Here's a fact. You are not going to earn a spot on this table with a certificate. The path to this table is through credibility and reputation.

My first bit of advice is if you want a worthy job, then you need to be worthy. Examine the work that you have done to date. Does it reflect your skill and perhaps more importantly, your ability? What about your attitude? Does your work reflect that of someone who is passionate and who loves testing? You don't need rockstar status, but you do need to be an eager apprentice. If you have aspirations to get a great testing job, but you're not prepared to put in the hard work, then why should you deserve a great role?

Here's something I do that has proven to be very useful. When I start a new role, I ask myself two questions. The first is "If I leave, how do I want people to remember me"? And second is "what legacy do I want to leave behind me"? It may sound ruthless to think about an exit strategy when starting a role, but the reality is, NO job is permanent so why treat it as one?

So, you are now a worthy tester, the next step is to be able to demonstrate this worthiness and please, put away that tired old resume! I'm talking about blogging, and speaking and contributing to the testing community. Contributing to the community is a great way of meeting local and international people and you learn so much. Regarding speaking, this doesn't have to be large conferences; there are plenty of small local meetups that offer you a space to speak. No tester meetups near you? Why not create one, or speak at a developer meetup.

Thirdly you need to network. I can't emphasis this enough. This is where the jobs are. You need to consider two types of networking, local and online. Local is essential if you want to find work in your

area. This means meeting people face to face at the local meetup. Yes, I know masterchef is on a Tuesday night and this clashes with the meetup, but hey, do you want a great job or not? Online networking is important too because it allows you to connect with like-minded people, plus it's a great source of learning. Many jobs come from both online and local networks.

You also need to research. Find out the good companies, speak to people through your network (not agencies) about the 'good places' to work, and make a plan on how you are going to work there. Having an online presence helps a lot here, but so does face to face networking. And be patient, great jobs don't just drop off trees and fall into your lap.

Yes, ultimately, these jobs don't come easy. They require hard work, and a willingness to put yourself out there. It comes at a cost to your personal lifestyle. But hey! It's all about choice. Great jobs are around, but it's about seizing the day and making the opportunity instead of relying on an agent to do it for you. This is a good thing. Trust me. As I said at the start, why should you put your fate into someone's hands?

The good news is more than ever before, companies who recognize and value their staff, who recognize and value quality testing, are recruiting in a grass roots way. If you want these types of jobs, it's easier to get them.

Now maybe this all seems like too much work and you know? I can live with that! Seriously, it's your call. But don't tell me that certificate is mandatory to work in testing, because it's not true. Many companies who 'understand' testing will hire you without a certification.

The question that is probably more pertinent is: "Do they want you?"

So get out there, work your butt off and then market your fabulous testing skills. If you stop putting your pearls before swine, one day that dream job will be yours!



Anne-Marie Charrett is a testing coach and trainer with a passion for helping testers discover their testing strengths becoming the testers they aspire to be. Anne-Marie is director of Testing Times, a company that offers coaching, training and consulting on software testing.

She specializes in test team transformation, creating and improving test teams to become powerhouses of testing excellence. She does this by focusing on a tester's skill and empowering them to make the changes themselves, securing long lasting change.

Anne-Marie can be found on twitter at @charrett & blogs at <http://mavericktester.com>.



There was a time when people did not have compass to find right direction. The only guide they had was that guiding star up in the sky.

Do you think that you are also stuck somewhere with technical issues? Do you need help in decision making or want guidance?

Well, the wait is now over . Introducing...

“The Guiding Star”

*The panel of our experts is now here to help you.
Send us your questions around software testing and our Guiding Stars will help you out.*



E-mail your question on –

theguidingstar@teatimewithtesters.com

Please Note :

1. This is not a job portal.
2. Typical interview questions will not be answered.
3. Questions should be on Software Testing or related topics only.



12th International Conference on Software QA and Testing on Embedded Systems

**MORE
TESTING**

**MORE
QUALITY**

**MORE
EMBEDDED**

This year we offer a special program: **3 Keynotes, 2 Tutorials, 8 Tracks**, from renowned companies. Visit our website -www.qatest.org- to **discover the Programme!**

QA&TEST, the international Conference on Software QA and Testing on Embedded Systems, will be held on 29, 30 and 31 October in Bilbao, Spain, and gives you an excellent opportunity to increase your business network and establish contact with others professionals of the industry.

The main objective of QA&TEST is present the last technological developments in Software Testing and Quality Assurance, and showcase successful best practice to reduce costs and give companies a lead in global competition.

www.qatest.org

SOFTWARE

Organiser



Sponsor



October 29 · 30 · 31

2013

Bilbao · Spain

In the school of Testing

for your better learning & sharing experience

Addressing the



Of Exploratory Testing

part 2

- by Leah Stockley

In my experience, by far the biggest risk to successful implementation of Exploratory Testing (ET) is the complete misunderstanding of what is meant by the term. In this series of articles I share the concerns that have arisen whilst implementing ET and some potential solutions to overcome them.

In the first article I addressed how to help the test team gain the confidence to Exploratory Test. Once teams understand that ET allows them to be better testers, the next concern raised is that their project team would never allow them to implement ET. This second article will address:

How to gain the confidence of your stakeholders

Firstly, by Stakeholders I mean 'Project Managers, BA's, Developers, CEO's, Users ... anyone who has an interest in the testing or can make a decision which affects it. And before I go on, let me share how not to do it. If you tell your stakeholders "We're not going to write test scripts anymore. They take too much effort. Instead we're going to exploratory test. I read about it on the Internet and it worked for Microsoft" they may actually hear "We aren't going to produce any documentation.

You will no longer be able to track our progress. We're cutting corners and removing process. Just trust us". Unless you have a laidback stakeholder, this approach is unlikely to convince them that implementing ET is in their best interests.

Prepare for the discussion

Dale Carnegie gave essential advice for influencing stakeholders *"Talk in terms of other peoples wants"*. In other words, tell them why **they** would want you to use ET and **how it will benefit them**. In order to do this successfully, I start by asking myself the following questions:

1. "What do my stakeholders expect from the test team?"
2. "What is important to my stakeholders?"
3. "What might be their concerns when introducing ET?"

"A Tester's job is to produce test cases"

The common answer to question 1 is "We expect the testing team to produce and execute test cases". Whilst this expectation may be true, we need to dive deeper and clarify the underlying intention behind the statement. In reality stakeholders 'demand' test cases because that is how the industry has taught them to manage the test team. Our job is to help them understand what they actually need from us and re-educate them on how best to manage us so we can deliver to their expectations. When you drill down, the underlying expectations look more like 'Prevent users from finding issues when the software is live' or 'Help deliver better quality code to UAT'. Production of test cases gives no guarantee of either of these. You may or may not have this exact conversation with your stakeholders, but it is important that, as testers, we understand our fundamental purpose on the project.

ET can be all things to all projects

Question 2 reminds me to consider what is important to my stakeholder. Often when we discuss ET we assume the benefit is saving money or time. I once based a presentation around that assumption. I got half way through before hearing "I don't care about saving time or money. I care about the quality of the product". Thankfully I also had a good case study to demonstrate how ET can address that too (a project that increased from 2 weeks to 6 months test effort, because the improved analysis required for successful ET meant they could communicate the impact of the proposed change and therefore clearly show the risks and the test effort required to mitigate it). In order to avoid this mistake, I now directly ask the stakeholders what is important to them.

Start off on the right foot

Question 3 is perhaps the most powerful. I put myself in my stakeholder's shoes (easier if I know them, but still possible if not. I find someone who does know them and can give some insight into their character/ work style). I then consider what is most likely to scare them and cause them to reject the proposal ... knowing this, I can start the conversation with a focus on aspects they will identify and agree with. If they have a leaning towards Agile processes then I can focus on the efficiency of less documentation & the collaboration gains. But if they are more conservative and enforce rigid process & documentation I will discuss 'a new structure' and give examples of the documentation we will produce. Ultimately I am describing the same approach to both, but tailored to the needs/ wants of the audience to show I can anticipate and address their concerns.

So how does this information help to gain Stakeholders confidence in ET?

I recently presented to a team who are notorious for adhering to strict processes. I knew that starting the conversation with a focus on less documentation would immediately alienate them. Instead I began by explaining **why** it was important to change our existing process (because it was preventing us from delivering to their project's expectations of timescales & quality). I said I would like to suggest a 'New Structure' that would allow us to deliver successfully. Once I had agreement on the reason behind the change, and confirmed I was refining the structure, not taking it away, the stakeholders were very open to listen to my ideas. I then used reassuring terms such as SET "Structured Exploratory Testing" (See part 1 of this series in last month's TTWT) and showed how we would map test coverage and agree priorities with them using Visual Test Models (see my recent blog post <http://lnkd.in/hxhwpm>).

I explained how we walk stakeholders through high-level test scenarios, how we provide auditable test evidence by taking detailed notes as we test. I showed an example of how we report progress & quality. I showed how their concerns would be addressed (test coverage, planning, audit requirements & progress reporting) in a positive structured way before they even raised them.

Explain the benefits for them, not for you

From experience, I can guarantee if I started the presentation by saying we want to use exploratory testing and would no longer write test scripts, I would have been on the back foot and defending my position from the start of the meeting. Instead I gained the project's agreement because I helped them understand how SET would benefit them. If I had told them we want to use ET because the traditional way wastes time creating scripts that never get followed to the letter, and causes us much maintenance/rework... that audience honestly wouldn't have cared because that wasn't a frustration or concern they could relate to.

Reporting test progress to Stakeholders without test cases

For many years now, the test case pass/fail/no-run count has been the de-facto measure of completion of testing. We are conditioned to produce the numbers and the stakeholders are used to looking at them to interpret our progress. So how can we replace this familiar structure?

Start by asking yourself 'What information am I trying to ascertain from a test case count?' In majority of cases it would be

- Are we on schedule? (Total executed vs. total planned)
- How important is the feature? (More test cases = more effort = more importance)
- What's the quality of the feature? (How many have passed/ failed)

Now ask yourself whether one number can accurately represent all of those pieces of information? Is there a potential for that number to mislead people? Have you ever had to placate the project because on day 2 of a 10-day project you are not 20% of the way through your test cases? I have on many occasions. As testers we understand that the first few days will be slower due to environment troubleshooting or the discovery of more defects. Wouldn't it be good if we could prevent those phone calls that start by defending ourselves, that waste everyone's time? To report in a way that makes it very clear when the reader should or should not be concerned.

Fact based test reporting

These thoughts inspired me to redesign our standard test progress report. Figure 1 shows an excerpt from it. We now provide explicit information where previously we used numbers. We walk stakeholders through the new format before we start using it and agree the definitions of Importance/Quality with them.

Function	Module	Progress	Importance	Quality	Comments
Login	Login	Completed	High	Amber	5 x medium defects
	Forgot Password	In Progress	Medium	Green	
Open Account	Personal Details	Delayed	High	Amber	
	Credit Check	Blocked	High	Red	Defect-123 blocking testing
Trading	Buy	In Progress	High	Green	
	Sell	In Progress	High	Green	
	Amend	Not Started	Medium		
Reporting	Client	In Progress	Low	Green	
	Regulatory	Delayed	High	Red	3 x showstopper defects
Regression		Not Started			

Figure 1 Test Progress Report

We actually considered (at length) the inclusion of a %complete column to appease those who like numbers. But realised if asked to explain how we derived that percentage we would be unable to (with or without test cases, as not all test cases are equal). We considered developing a sophisticated risk & complexity weighted measure to more accurately produce the number, but agreed the effort involved to make that valid and reliable would far outweigh the benefit of including the number in the report.

How we report testing & quality is a conversation we should be having with our stakeholders, regardless of whether we write test cases. We can unintentionally mislead them by relying on numbers and graphs to tell the story. So we took the decision to roll this report format out to all projects, even those who still use test scripts. Overwhelmingly the feedback has been positive, because they now receive information that clearly shows where the problems are.

Removing the Blockers for ET

Understanding how to report progress is a major blocker to some projects implementing ET. Having removed a reliance on test case counts from reports, stakeholders more readily understand how testing can add value without test cases. It allows them to focus on the important information the test team provides... clear information about the status of risks within the product.

This makes future conversations about Exploratory Testing much easier because the underlying reason for change is the same "We focus on providing you important information on the quality of the software we are testing, not the quality of our testing process".

Understand why you want to make changes

Some people thrive on change, while others run away from it. The remainder might follow when someone else has proven it will work. Fundamentally, when considering implementing a change, especially to an 'established' process (i.e. test scripting, test reporting) I consider the questions:

What is the underlying purpose of the current process? (When I've answered I dig deeper by repeatedly asking myself "And what does that give me?")

What are the problems / frustrations with the current process?

With these questions answered, I am better able to design a suitable process and understand how to articulate the need for change to others. When combined with the 3 questions above, I'm also able to explain the benefits in their language. By focusing the conversation directly on the Stakeholders needs we are better able to anticipate their perceived risks, alleviate concerns and demonstrate the benefits of implementing Structured Exploratory Testing.

Having addressed the concerns which arise both from the tester and the stakeholder's point of view, the next article will focus on addressing the concerns of test managers themselves when starting to use Exploratory Testing.



Leah has more than 14 years experience in Software Testing for consultancies, banks and software houses across multiple industries.

Her Context-Driven awakening began 2 years ago. In that time she has become passionate about inspiring and coaching testers to be innovative, empowered and to continually improve the art of testing.

Currently working at a large global bank, Leah is responsible for rolling out the CDT approach across the Global Test Centre and developing solutions to the concerns that arise when introducing & applying CDT.

When not at work, Leah shares her experiences on www.inspiredtester.com

[Back To Index](#) 

A hand is shown holding a red stick figure, which is positioned above two groups of blue stick figures. The blue stick figures are arranged in two pairs, each pair holding hands. The red stick figure is being held by the hand, as if it is being introduced or selected. The background is white.

Hiring **Great** Testers

[A How-to Guide]

by Johanna Rothman

Interviewing to Detect Great Testers

Now that you've thought of your culture, back in *What Makes a Great Tester*, and you've considered *How Technical Your Tester Has to Be*, it's time to think about how you would detect a great tester. That means it's time for an interview.

You've read through the resumes, discarding the ones that seem hopeless. You've phone-screened the candidates and you are ready to interview the candidates who seem to fit your culture and who might have the technical skills—and more importantly, the cultural fit to work with you.

How do you create a great interview for your potential tester?

Interviews Are Conversations

The best interviews are two-way conversations, not interrogations. If you have a respectful conversation, you will learn about your candidate. And, your candidate will learn about you and your organization. You don't have to sell the candidate on your job or your organization. Your questions and your interview style will do that for you. You can focus your interview on learning about the candidate.

Back in *What Makes a Great Tester*, I asked you to consider your culture and potential tester qualities, preferences, and non-technical skills. In *How Technical Does a Tester Have to Be*, I asked you to think about the four dimensions of technical skill. Now it's time to put it all together to create your interview of great questions.

Determine the Essentials

You have plenty of options. Select the essential cultural, non-technical qualities, preferences, and non-technical skills. Now, select the essential technical skills. Those are the areas you want to focus on for the interview.

Let's imagine for our tester that we want to focus on these essentials:

- Someone who is great at working in a cross-functional team
- Someone who can advocate for specific defects
- Someone who can learn quickly
- Someone who is highly adaptable
- Someone who has enough code-writing skill to write automated tests in our test tool
- Someone who is adept enough to recognize when we need to do exploratory testing
- Someone who can learn the solution space to explore deeply, both in automation and manually for the best testing bang for our buck

This is not a junior tester. This is a mid-level to senior-level tester. I might also expect some coaching of other testers with this person. But I have the essentials now. I can create the interview matrix and create some questions and an audition.

Develop the Interview Matrix

When you have a candidate visit you for an interview, make sure everyone on the interview team does not ask the same question. The best way I know to do that is to create an interview matrix with everyone across the top and the areas down the side to make sure you have covered the areas in this interview day.

Interviewer/Area	1	2	3	4	5
Teamwork	x			x	
Learn quickly		x			
Test automation			x	x	
Bug Advocate	x				x
Adaptability		x			
Exploratory testing				x	
Learn solution space			x		x

Don't Ask These or Other Irrelevant Questions

Interviewers ask these questions all the time. They don't tell you how the candidate will work at work. Don't ask them.

1. "How would you move Mt. Fuji?" Unless you have to move Mt. Fuji, don't ask this question.
2. "What are your weaknesses?" I can make any weakness sound like a strength. So can any candidate past his or her second interview with this question.
3. "What are your strengths?" I can sound like Superman or Superwoman with this question. Come on, we want to go to work, not save the world, here.
4. "Where do you want to be in five years?" Unless you are offering a five-year contract, don't ask this question. No one can predict that far in advance, and that's a crazy question.
5. "What do you do outside of work?" In the US, it's illegal to ask this question. Even if it's not illegal to ask this where you work, it's not a good idea. Why? Because you won't learn how the person works.
6. "If you had a magic wand..." Stop right there. Unless you have magic skills and are willing to teach the candidate, don't go there.

Ask questions that are relevant to the job.

Ideally, each interviewer would make sure to cover just two areas, and would make sure that each area was covered by no more than two people. This list is too large to be covered in one day with just interview questions. The way to handle this is with auditions, which I'll discuss in the next section.

Detect Great People with Great Questions

The way you separate people who don't fit your organization is by asking them questions, or by asking them to perform an audition—watching them work.

Closed questions establish the facts.

Open-ended questions that help candidates explain their stories.

Behavior-description questions are the best type of open-ended question. (Tell me about a time when...)

Auditions are a technique to see the candidate perform work, whether that work is technical or a series of behaviors.

Hypothetical questions help you see how a candidate thinks about a problem, or at least, how candidates *think* they think about a problem.

Meta-questions are questions about the interview or question. "Is there something else I should be asking you?"

Auditions show you behaviors. They allow you to see the candidate at work. I much prefer auditions to irrelevant or hypothetical questions.

Behavior-Description Questions Are the Best Questions

If you need to pick one type of questions, use behavior-description questions.

Below are some examples of behavior-description questions for each of the areas above:

- Tell me about your work on a team on your most recent project.
- Give me a recent example of a time you had to learn quickly.
- Tell me about your recent test automation.
- Have you ever had to advocate for a bug to get fixed? What happened?
- Have you ever been on a project where things did not go according to plan? What happened?
- Do you do exploratory testing? When? Tell me about a recent example.
- When was the most recent time you had to learn the guts of the system?

These are all examples of behavior-description questions. Since you and another person would ask questions, you would have to vary the questions. They are starting points for a conversation.

For this candidate, let's say you want to explore the topic of teamwork in depth. You might start with the behavior-description question, "Tell me about your work on a team on your most recent project." The candidate replies, "I was on an agile team. It was great."

That's not an in-depth answer, is it? You need to probe further. Ask a question like this, "Great! Did you develop automated tests during the iteration?" This is a closed question to establish the facts.

The candidate will reply with either a yes or a no. If yes, continue with, "How did you know what tests to develop?" That is an open-ended question that should prompt the candidate to provide a more in-depth answer. After that question, you might ask about acceptance criteria for features. You proceed, asking more and more in-depth questions about the most recent project.

Why the most recent project? Because the candidate's most recent experience is the best predictor of future behavior. It's not the only predictor, but it is the best.

Remember that the candidate might have replied with a no to the question about developing automated tests during the iteration. If the candidate did, you can ask a question such as, "Hmm, that must make it difficult to keep up with the developers over time. What did you do?" That's another open-ended behavior-description question. You want to hear what the candidate did, and not pass judgment on the candidate's company or team.

Auditions Allow You to See a Candidate at Work

Just asking questions is not enough. You want to see a candidate at work, too. Auditions are a great way to see candidates at work. First, you need to define the behaviors you want to see in an audition.

You might want to see a candidate perform some exploratory testing. In that case, you would provide a candidate a product, some time, a way to record defects, and a place in which to work. Make sure this is not a closed problem—that is a problem with only One Right Answer. You want a problem with multiple correct answers.

For example, you might ask a candidate how they would test a common browser or a web-based application, if you test applications such as browsers. If you don't test web applications, that would be a poor audition. Or, you might ask a candidate to test your application. You don't have to intentionally add problems to the application. Because your candidate is unfamiliar with it, I suspect your candidate will find problem handily.

If you are looking for a performance or reliability tester, you may have to help the tester instrument the code, if you want an in-depth test. You might even need the tester to sign a Non Disclosure Agreement

before looking at your code. But the problem with these kinds of auditions is that they tend to be quite long. I like auditions that you can time-box to about 20 minutes in duration. Otherwise, the audition feels as if you are taking advantage of the candidate.

You might want to test the audition with your testers to make sure someone can complete the audition in a reasonable amount of time, before you spring it on a candidate.

What Else Should I Ask You?

At the end of the interview, I like to ask the question, "What else should I ask you?" I want the candidate to tell me anything else I should know. I want the candidate to be able to "sell" him or herself.

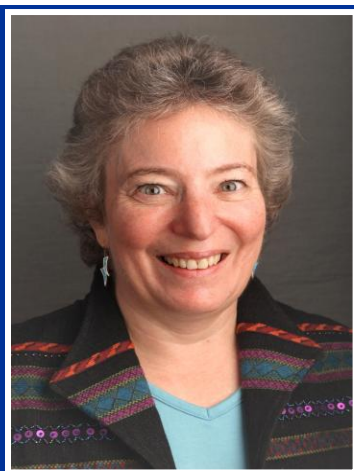
Sometimes candidates have more to say. Sometimes they don't. But I always want to give them a chance.

What Questions Do You Have For Me?

Leave time at the end of the interview to ask the candidate, "What questions do you have for me?" This will help you discover more about the potential cultural fit between your organization and the candidate.

Savvy candidates take advantage of the end of the interview to ask questions of the interviewer. Candidates who are new to their job search might not remember to do so.

Always answer the questions honestly. Remember, the interview is selling the organization.



Johanna Rothman is the author of *Hiring Geeks That Fit*, <https://leanpub.com/hiringgeeks>.

See her other books at <http://www.jrothman.com/books/>.

She writes an email newsletter, the Pragmatic Manager, <http://www.jrothman.com/pragmaticmanager/>

[Back To Index](#) 



Taking
a break?

a click here
will take you there

WORLD CONFERENCE — NEXT GEN TESTING

08-12 July, 2013 | Le-Meridien, Bangalore

register@nextgentesting.orgRex Black
Keynote Speaker

UNICOM's Next Generation Testing Conference has been the most successful and widely acknowledged gathering of Software Testing Professionals. In 2012, the Next Generation Testing Conference took place in London, Bangalore, Mumbai, Chennai, New Delhi and Colombo. With over 850 attendees participating in India last year, it proved to be a highly informative and innovative event for software testers.

UNICOM is organizing "**World Conference - Next Generation Testing**" in Bangalore from **08-12 July 2013**. The first three days will consist of workshops presented by world acclaimed testing experts, and these workshops provide the highest standards of training available. Workshops will take place from Monday to Wednesday from 8.30am-5.00pm. The conference and exhibition takes place over the following 2 days on Thursday and Friday.

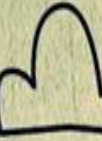
This conference is expected to have over 500 delegates, 45+ presentation, 25+ speakers, 10+ Workshops, 4 Parallel Tracks.

To view the Program& Workshops [click here](#)

To Register [click here](#)



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



then you must be telling your friends about ..



Tea-time with Testers

Don't you ? 😊



Tea-time with Testers !

first choice of every #smart tester !



testing intelligence

- *its all about becoming an intelligent tester*



an exclusive series by **Joel Montvelisky**

Coordinating your automated and manual test management

I received great response on one of my past articles i.e. "Manual and automated tests together are challenging."

To summarize the article, it talked about the fact that some QA teams that have automation in place still run part of their automatic test cases manually.

WHY do they do this? Some of the reasons I got were that (1) they like running some of their tests manually as well since they are "really important", (2) that automation cannot be trusted 100%, (3) there is no clear definition or knowledge of what is exactly automated, and (4) they run test manually in order to update their test management system with the results (of their automation) and be able to generate complete and comprehensive reports for their management.

Some more reasons why to manually run your automatic test cases:

So as I said, I got additional feedback from a number of people who read the post (thanks to everyone who got back to me!). Some of the most interesting points raised were the following:

1. Even though the test may look similar at first sight, they are testing different aspects of the same feature. For example an automatic test may verify the installation procedure on all O/S, and a "related manual test" will verify the effects of abnormal network traffic during the installation process.
2. There are some tests that are "semi-automatic" and not fully automatic, like blink-tests where you perform a number of operations and constantly take screen-captures of the application that are then reviewed quickly by the tester in order to detect GUI issues or bugs.
3. You can start additional manual tests on the places where your automation concluded. For example you can use automation to check the installation of the system and to validate the procedure to add, remove and modify data. Once these tests are done you can take the system created and continue running additional manual tests based on existing data and system setup.

Additional challenges when coordinating manual and automatic testing

Part of the feedback I got to the post was also in the way of additional challenges that rise in coordinating the work of your manual and automatic testing (not directly related to re-running your tests).

Here are some of the most interesting points:

Figuring out what needs to be moved from manual to automatic testing

This is the first challenge and I guess one of the most significant decisions you need to make, since it will dictate the value you realize from your automation effort. I am not sure there is a text-book answer to this question, but I guess it needs to be related to 3 central factors:

ROI – the return on the investment to automate your test, or in plain English what will you gain from automating this specific script. This is usually a matter of how long it will take you to automate your script vs. how many times will you run it automatically and how much time do you save because of it.

Complexity of the Test – this refers to both how hard it is to create the automatic script but also how easy it is to do the test manually. There are tests that are simply too hard to do manually (e.g. testing the API of your product) and there are others that are practically impossible to automated (e.g. usability of a feature). You need to choose correctly what tests are more suited to be either automatic or manual.

Stability of your AUT – maybe the biggest challenge of any automatic test is to cope with changes on your AUT (Application Under Test). There are some automation platforms that do this better than others, but no automatic system will be able to "improvise and understand" in the way human-testers do. Since this is the case you need to add this factor to the list of things that will help you choose what to automate, and try not to work on areas of the product that you know will go through radical changes in the near future.

Coordinating the tasks between the manual and the automation testing teams

These are also big challenges that arise from the intrinsic differences between a good manual test engineer and a good automation engineer.

Who does what?

- Do the automation engineers define what to test or is this the call of the manual testers?
- Is scheduling tests a task of the manual testers or the automation testers?
- What happens when an automatic test fails, is the manual tester in charge of verifying if there is a bug or is this the job of the automation team?

All these questions and many more like them need to be defined up-front, and again here there is no text-book answer.

Personally I believe that a good automation engineer is closer to a developer than to a tester and so I like placing more weight on the tasks of my manual testers. Basically I ask my automation engineers to be "service providers" and to work with my manual testing engineers to supply them with the best possible automatic answer to their testing scenarios.

I also believe that automation is part of the complete testing effort and so the decision and responsibility of what to run and when to run it should be in the hands of the group in charge of the complete testing efforts (this is usually the task of the manual testers too).

Lifting the taboo from test automation

I really liked this comment from Marco Venzelaer in LinkedIn: "Some automation teams make test automation a 'dark art' which is closely guarded within the team..."



This point got me laughing but also grabbing my head as he managed to articulate something I had felt for a number of years. Some automation engineers feel that by treating their work as something mysteriously hard and extremely complex they gain some sort of work security. They are especially successful by seeding these feelings on manual testers, who don't have any coding experience on their side and so believe what their automation colleagues are telling them.

Other than a false feeling of job security, these automation engineers also manage to generate a sort of

taboo around their work that will make manual testers refrain from trying to understand it or have any influence on the process. In the end this behavior is harmful to the coordination of efforts and has a deterrent effect on the **overall achievements of the testing team**.

So how do you coordinate the work of you automated and manual testing in our team?

I think that the key to maintaining a healthy manual-Vs-automation relation in testing is **TEAMWORK & COMMUNICATION**

You need to make sure both teams, the manual testers and the automation engineers, are working based on the same agenda, coordinating their tasks based on the same priorities, and understanding how each player helps the other to fulfill the goals of the organization. In simple terms, how they work together in order to make their testing process faster, broader and eventually more effective.

Important things to take into account are:

- 1. Make sure they are coordinated**, by having regular update meetings and making sure there is active participation of both teams when planning the testing tasks for each project.
- 2. Have both teams work on an integrated environment** where both manual and automated tests are managed and executed. This will allow all your testers, and also every other person in your company, to see the "full picture" and understand both on a planning and execution level what is been covered, what has been tested, and what are the results of these tests. After all no one really cares if the test was run manually or automatically, they care about the results of the test.
- 3. Have a process in place** where automation is a tool to help your manual testing team. The correct process is what will eventually make or break the relationship between manual and automatic tests. Both teams need understand that the idea of automation is to free the time of manual testers to run the more complex tests, those that are hard or expensive to automate. Once they understand that they complement each other and not compete with one-another they will be able to focus on their shared challenges instead of their rivalries.

In short you need to make sure your teams have both the agenda as well as the infrastructure required in order to coordinate their work.

In the end, it is the work of the manager and the Organization to create the environment and the "team rules" that will allow all members to feel like they provide their work and help drive the project forward TOGETHER.




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

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

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

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

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



Call for Articles !

Have you got something to say?

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

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

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

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

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

T ' Talks



T. Ashok exclusively on software testing

The onion peel and the half-filled tea cup.

Good understanding is critical to good testing. So how does one understand a system well quickly? The typical belief is that prior experience and domain knowledge play a key role here. But what if we do not have sufficient prior experience in that area? That is when we like to resort to creative thinking, good questioning as some of the means to understand. Now comes the question - "how does one come up with good questions" to understand better? In my discussions with folks in the community, the typical answer I get is that this is based on prior experience. Now we are back to square one- Experience! My take is that there is a logical/scientific way to go about this and therefore need not be driven only by experience/domain-knowledge. Let us discuss this in detail...

The act of understanding is interesting! It is immersive, non-linear, instinctive, frustrating yet fun, detailed but not forgetting the big picture, time constrained and extremely satisfying when we "get it". To look at the act of understanding in a scientific manner, it is necessary to be clear as what we want to understand and when. How do we onion-peel?

End users their expectations
Business flows & technical features
Interactions & linkages
Key attributes expected
The deployment environment
Architecture & technology
Conditions that govern behaviour
Inputs - Specification & interface

Using the picture alongside, Let us start from the "outer layer" of the onion peel (top). It commences with understanding who the consumers/end-users of the system are and how the system is intended to help them, what they may expect of the system. Thinking from the end user perspective, the focus is to understand what each of them do/intend-to-do with system and build the baseline of the use-case(s) /business flow(s).

Now we are ready to identify the technical features that enable these business flows to be accomplished and then setup the technical feature baseline. Having broken down the system into constituent flows/features, understand their "connections" I.e. how does each flow/feature depend on other and therefore their linkages.

Before we get deeper into understanding in detail how each flow/feature works, understand the key attributes of each of them. Note that we have decomposed the system and setup a clear baseline of the system in terms of users, their business flows, constituent technical features and the associated attributes. Now

we are ready to delve into the details. Did you notice that we have minimized our dependency on domain knowledge ?

Let us peel the onion further... Before we get into the detailed understanding of each flow/feature, understand how the system is deployed. What other systems does this interact/use, how is it deployed, what is the environment in terms of hardware, software, versions this needs to support/uses. Having understood the external environment, delve next into the internal structure - what are the various constituents of the system, how are they interconnected, the interfaces and the technologies used to construct these. What we have done is to get a good feel of the external environment and the internal structure. Now we are ready to understand the behaviour of each flow/feature.

What is behaviour? It is about transforming the input(s) to appropriate output(s) to outputs based on certain conditions. So all we need to do is understand for each flow/feature, the inputs and the outputs, their specification and the conditions that constitute the business logic. Finally we need to get understand how these inputs enter the system i.e. input interface.

Good understanding is not about knowing everything in detail, it is about just knowing what is required and learning to discard what is not necessary or deferring to a later stage.

I am reminded of a Buddha story, listen to this.

Once upon a time there lived a wise and learned man who had mastered various religions, philosophy. He was proud of his knowledge and wanted to learn more. He came to know that Buddha who lived in another state was a very learned man too. He was keen to further his knowledge and went to Buddha, introduced himself as a very well read and learned man who know all the systems of religions & philosophies and expressed his wish to enhance his knowledge and requested to teach him. Buddha said he would be happy to do and requested him to kindly sit in the corner of the room. A few hours passed and the learned man was getting fidgety while Buddha continued to do his work, and as the sun went down Buddha requested that he will do so next day. The man went away only to be back next morning, Buddha requested him sit down in the corner. The morning turned to afternoon and dusk was setting in. The man was thoroughly upset that Buddha was ignoring him, he went up to him and said "I am upset that you are ignoring me, I hope you understand that I am a learned man and posses a lot of knowledge

in the matters of religion, philosophy". Buddha was expecting this and he said "Come, sit down let us have tea". Buddha took the tea pot and poured the tea onto the cup while watching the man. The cup was filled to the brim and Buddha continued pouring, spilling the tea. The man then shouted "The tea cup is full, stop pouring!". Buddha said "Your mind is like this, full with knowledge with no space to acquire the additional knowledge". Empty it and you will be ready to learn.

Good understanding requires to ensure that tea cup is only partially full i.e. know only what you require for now, defer what is not required so that you do not just know, you understand. It is not attempting to know all.

Before you jump into testing anything, Understand. Don't be fazed by complexity nor be worried by no prior knowledge. Apply the onion peel keeping the tea-cup half filled.

Enjoy!

[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



Testing PUZZLES

by Sebi



Claim your **Smart Tester of The Month** Award. Send us an answer for the Puzzle and Crossword bellow b4 5th June 2013 & grab your Title.

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

“Find it out”

Find the flash file (.swf) located on <http://paypal-education.com.au> and extract the AS3 code.



Back To Index



Biography



Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years.

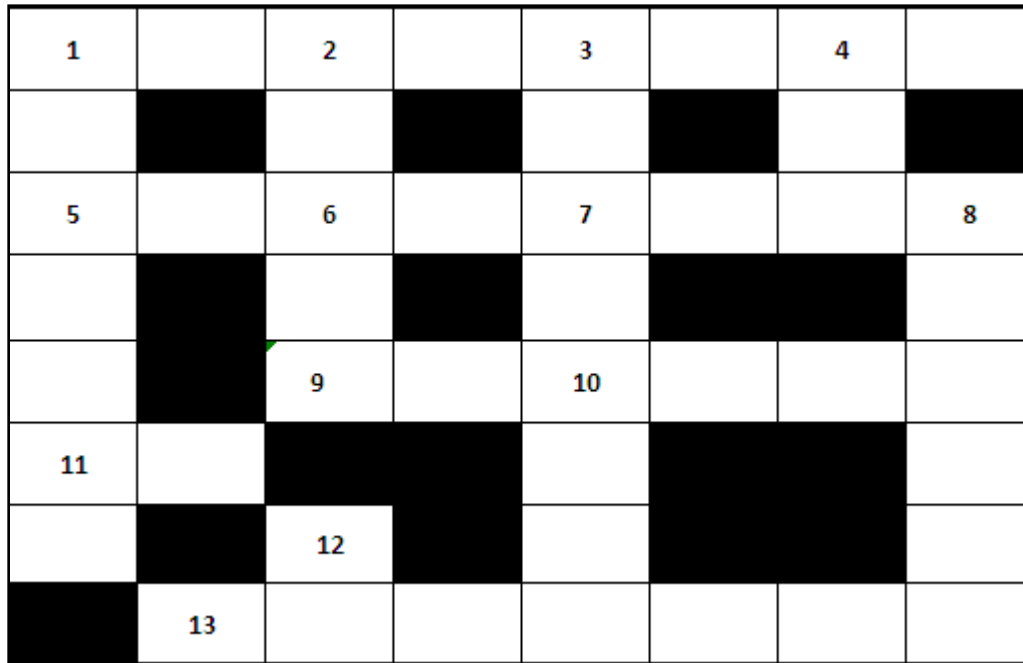
He considers himself a context-driven follower and he is a fan of exploratory testing.

He tweets as @testalways.

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



TESTING CROSSWORD



Horizontal:

1. It is a tool to conduct stress tests by eliminating inconveniences and providing integrated environments (8)
5. It is a low-overhead, non-invasive black-box test automation tool (8)
9. It is an alternative to traditional project management, typically used in software development. It helps teams respond to unpredictability through incremental, iterative work cadences, known as sprints (5)
11. Subjecting the program to heavy volumes of data, in short form (2)
13. It is a Test Automation Management tool with capabilities of running GUI and non-GUI test automation (7)

Vertical:

1. It is a framework for Behaviour-Driven Development (7)
2. It is a type of software testing that intends to ensure that changes affected it, in short form (2)
3. It is a testing aimed at showing software does not work. Also known as "test to fail", in short form (2)
4. A test case design technique for a component in which test cases are designed to execute representatives from equivalence classes, in short form (2)
6. A synonym for White Box Testing, is called _____ Box Testing (5)
7. This term refers to making software specifically designed for a specific locality, in short form (2)
8. It is a tool that automates module/unit testing of embedded software written in various embedded dialects of the C/C++ programming language (5)
10. It is the fastest test harness software to learn (3)
12. The short form of system under (2)

Answers for last month's crossword:

T	E	S	T	C	U	B	E
E		T			N		M
S	I	K	U	L	I		U
T			T		X	M	L
L	O	A	D	U	I		A
I		P			A	U	T
N	L	I	N	U	X		O
K		T		C		C	R



*We appreciate that you
"LIKE" US!*



Join us on Facebook.

You are just a CLICK AWAY

**Note- We have declared winners' names on our
Facebook page.**



Every Tester

who reads Tea-time with Testers,

**Recommends it to friends and
colleagues .**

What About You ?

in ne>xt issue

articles by -

Jerry Weinberg

Johanna Rothman

T Ashok

Joel Montvelisky

Ben Kelly

Leah Stockley

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Core Team:

Anurag Khode (Nagpur, India)

Dr.Meeta Prakash (Bangalore, India)



Anurag



Dr. Meeta Prakash

Testing Puzzle & Online Collaboration:

Eusebiu Blindu (Brno , Czech Republic)

Shweta Daiv (Mumbai, India)



Eusebiu



Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)



Kiran Kumar



Chris



Romil

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

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

Google™

Join our community on

facebook.

Follow us on



www.teatimewithtesters.com

