

THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

NOVEMBER 2013 | YEAR 3 ISSUE X

Jerry Weinberg

Managing Others – The Manager's Job

James Christie

Standards? We can do better

Mike Lyles

Are you Smarter than a Test Manager?

Jim Holmes

Preparing for Automation: Selecting the tools

Guy Mason

The Assumption Bias & Testing: How Does it Influence You?

JeanAnn Harrison

Rants and Ramblings of a Mobile Tester

T Ashok

Think better using "Descriptive-Prescriptive" approach

Participate in **The State of Testing** survey - 2013

Are you curious to learn more about the testing community? We believe that all testers are!

With this survey we hope to find answers of many interesting questions, such as:

- What are the main challenges faced by testers around the world?
- What does our professional environment look like?
- Where is the testing profession heading?

& even for things like **how tester's salary varies across different locations worldwide?**

QA Intelligence in association with **Tea-time with Testers** plan to conduct this survey every year, which will allow us to see not only a snapshot of our professional testing reality but also the trends in the field as they keep shifting year by year.

100s of testers have already filled this survey. What about you?

 **Take the Survey Now**





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



Testers and the State of Testing

Some weeks ago my friend, Joel Montvelisky (author of <http://qablog.practitest.com>) was looking for information to write a post about the advances in the testing world in the last 5-10 years and he realised that there is no centralized set of information that provides visibility into what is happening and what are the trends in the world of testing today. In principle he was looking for something similar to the State of Agile survey that most of us review each year when it goes out, and he was not able to find something that provided such information.

People often contact me to know the latest trends in testing field, tools and techniques in demand, market requirements and skills that they should develop etc. and hence when Joel contacted me, we decided to turn this into a project i.e. to conduct the **State of Testing Survey** that will provide a snapshot of the testing field's reality. It'll also help us to capture some of the trends as they shift year by year.

How can you contribute?

It's easy. We have already **launched this survey** and it will run for some odd 10 days. It's only you dear readers who can make this survey count and help us get maximum data so that we can aggregate and analyse it, only to present in front of you folks again. So, please spend your 10 minutes on this survey and help us in helping you yet better next time.

That's all for now. Enjoy the festival and don't forget that I'll be waiting for your response😊.

Yours Sincerely,

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



QuickLook

Participate in The State of Testing survey



Testing Puzzles
by Sebi

Crossword
by



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

Standards? We can do better – 13

Are you Smarter than a Test Manager? - 17

The Assumption Bias & Testing - 24

In the School of Testing

Preparing for Automation: Selecting the tools – 28

Rants and Ramblings of a Mobile Tester – 35

T ' Talks

Think better using "Descriptive-Prescriptive" approach- 41

Testing Puzzle – S.T.O.M. Contest

Family de Tea-time with Testers

What's making News?

- find out the latest happenings in the technology world

PER SCHOLAS GRADUATES OFFERED FIVE TECHNICAL TRAINING POSITIONS AT BARCLAYS

Bronx, N.Y. November 27, 2013 — Per Scholas, a national non-profit providing free IT job training and career support to individuals in underserved communities, announced today that five of its graduates were offered positions in a technical training program launched by corporate partner Barclays, the major global financial services provider. Five graduates from IT-Ready, Per Scholas' free job training program, began their positions in this two-year long program on November 18, 2013.

"Barclays has been an engaged corporate partner with Per Scholas over the past two years providing volunteers, funding, and board leadership," said Plinio Ayala, President and CEO of Per Scholas. "Since 1998, more than 4,500 adults have enrolled in our flagship IT-Ready job training in New York, and Cincinnati and Columbus, OH, with graduates earning collectively more than \$150,000,000. The commitment from Barclays to create opportunity for these individuals is a key contributor to this life-changing work."

Before starting at Barclays, selected students completed the free hands-on 15-week training program provided by Per Scholas, which includes the completion of the CompTIA A+ and Network+ industry certifications. The graduates will work in areas including end-user support, networking operations, and distributed technology support within Barclays.

"We are incredibly proud to partner with Per Scholas and are thrilled to work with their graduates," said Wayne Kunow, Regional Head of Global Technology Infrastructure and Services at Barclays. "The high-quality training the graduates receive compliments the areas within our organization where we are looking to hire. We look forward to helping the graduates become successful IT professionals."

Per Scholas works prominently in the Bronx serving residents from all the five boroughs, and is in the process of expanding to other areas around the country.

ABOUT PER SCHOLAS

Per Scholas is a 501(c)(3) nonprofit offering free, high quality technology education, job training, placement and career development opportunities to people in underserved communities. Since 1998, more than 4,500 un- and underemployed adults (18+ years old) have enrolled in its IT-Ready job training and placement programs in New York City, Cincinnati and Columbus, OH. Its fourth operation will open in the Washington DC region in early 2014. The Social Impact Exchange recently named Per Scholas one of the top 100 nonprofits creating proven social impact in the U.S.

###

Media Contact

Jessica White

Per Scholas

718-772-0623 | jwhite@perscholas.org

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

→ point blank ←

Just as the API Revolution has injected a new vitality and creativity into the software development process, the “Software Eating the World” concept has injected a new urgency and philosophy into the testing industry. We have some hard problems ahead of us and we need to find the right balance between speed, innovation, and quality – not an easy trio to manage.

Lorinda Brandon

Quality Evangelist



I spent hours searching for the cause of a bug where it was not. Discovering where it was not helped lead me to where it was.

Ben Simo

Software testing expert

Amazingly, people who say “Failure is not an option” are in fact selecting the failure option: by driving truth away.

James Bach

Software tester, author, trainer and consultant

The numbers are not telling you what to do. The voices in your head are telling you that, and they may be pointing to some numbers.

Michael Bolton

Software testing expert, trainer & consultant

Never lie, and never pay attention to those who say stupid things like Testing is dying.

Gerald M. Weinberg

American computer scientist, author and teacher of the psychology and anthropology of computer software development.

Tea & Testing



with

Jerry Weinberg

Managing Others – The Manager's Job (Part 4)

Helpful Hints and Suggestions

1. There is an interesting trade-off between trust and listening, as noted by M. DePree, in *Leadership is an Art*: we realized that, while understanding is an essential part of organized activity, it just is not possible for everybody to understand everything. The following is essential: We must trust one another to be accountable for our own assignments. When that kind of trust is present, it is a beautifully liberating thing.

2. Because of their position in the hierarchy, managers also make implicit assignments. For instance, every time the senior management team changes one of their meetings, the change ripples down through the whole organization.

One telecommunications company studied this effect using scheduling data from their e-mail system. One Vice President who changed the time of one meeting with her immediate staff led to an average of 670 E-mail messages being generated as the change rippled down. The study estimated an average work time per message of 12 minutes (including phone calls and personal contacts not recorded). This meant $670 \times 12 = 134$ hours of rescheduling time, at a burdened average cost of \$70 per hour yielded \$9,380 per Vice Presidential meeting change.

3. Your personal experience is never enough to manage a large project, because in one lifetime, you cannot have experienced enough different large project situations. How many five-year projects can you have completed in a twenty-year career? To be an effective manager of large projects, therefore, you must learn from others. If you don't do this well, then you must learn how to learn from others.
4. Dan Starr says giving the experienced people a strong-willed assistant runs afoul of the tradition of assistants being a "perk" and status symbol, not to be given to lowly techies. Thus, if you call them secretaries, you may not succeed. Ironically, you may be more successful implementing this technique by assigning a junior technical person who earns three times as much as a secretary.
5. Both Dan and Mark Manduke reminded me that firing the manager of an unsuccessful team is precisely what is done in sports. But even those managers who are fondest of using sports analogies are likely to protest this approach does not apply to them.

Summary

1. In an effective software organization, the manager's job is getting more people involved (and getting people more involved) in decisions about what is to be done, and in doing it.
2. For example, to recover from a crisis, a manager has to mobilize sidelined people in order to have the resources to keep the ship afloat.
3. As an organization moves deeper into crisis, the best-informed people tend to become overloaded. Managers may unknowingly contribute to this piling on, and can counteract it by congruent management action.
4. In Pattern 2 (Routine) organizations, managers *prescribe* the way things should be done, rather than *describe* what outcomes are desired (a more Pattern 3 behavior). Managers who prescribe tend to listen to employees in a super reasonable or blaming way, without hearing them.
5. A major job of the software engineering manager is to develop openness and trust among all the workers who are contributing to successful software activities. This requires honesty and true listening.
6. Right or wrong, the reasons given by employees always contain the information an effective manager needs to steer the organization.
7. The Pattern 3 (Steering) manager does not generally evaluate quality, but establishes the processes, not people, to evaluate quality. But when the manager *does* evaluate the quality of work, the main purpose of the evaluation is to help the employees develop their own skills.
8. Managers who believe the One-Dimensional Selection Model have no room in their repertoire for coaching, teaching, or training.
9. Congruent managers are not locked into blaming, partly because they know they are not passive victims of their employees, their bosses, or the dynamics of software quality. Instead, they use all of these factors intelligently as resources.
10. Leadership is the ability to create an environment in which everyone is empowered to contribute creatively to solving the problems. In this model, the manager's job may be evaluated by one and only one measure: *the success of the people being managed*.

Practice

1. If you are a manager, use the list in Section 6.7 to survey the people who work for you. Ask them if any other things should be on the list of good things you do as their manager. Use their responses to guide your behavior.

2. If you have a manager, use the list in Section 6.7 to study the style of your own managers. What would you want to add to the list of good things they do? What things missing from their list would you like them to do? How would you go about getting them to do those things?

3. How important is software experience in management? We've seen technical experience can be a handicap, but some can overcome it. Hardware experience can also be a handicap, or any experience in related areas that tempts you to drop down a level and do the work your employees should be doing. Get together with a few friends who have strong technical experience and share tactics you use to prevent your experience from reducing your effectiveness as a manager.

4. Here are four different models of the manager's job. The manager's job is:

- a. to do the work
- b. to make decisions about the work
- c. to hire and train people to do the work
- d. to hire and train people to make decisions about the work

Where do you stand on this question?

5. Some texts say the manager's job is to make decisions, but others say the manager's job is to avoid reaching decision points, such as having to fire someone. Where is your opinion on this question? How does it relate to software cultural patterns?

6. Discuss the following note Steve Heller put on the software engineering forum:

I couldn't agree more that the software industry has a very poor record of selecting the people to be promoted to manager. Partly, though, the problem is the Peter Principle: if you are doing well, you get promoted to the next position, so that you eventually reach your "level of incompetence," and there you stay. My defense against that is to stay technical, even though I have done some management on occasion and successfully at that.

How does Steve's position and career compare with your own? Have you ever experienced the Peter Principle in action?

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](#)

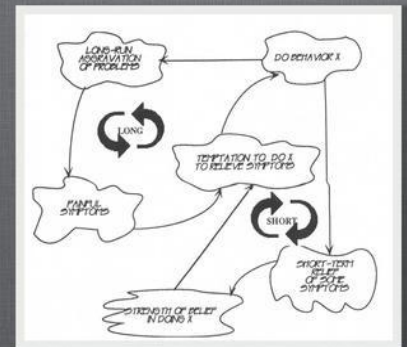
MANAGING YOURSELF AND OTHERS is another famous book written by Jerry.

Becoming an effective manager is the subject of this volume in Gerald M. Weinberg's highly acclaimed series, *Quality Software*. To be effective, managers must act congruently. Managers must not only understand the concepts of good software engineering, but also translate them into their own practices. Read this book to find out more.

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

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

MANAGING YOURSELF & OTHERS



GERALD M. WEINBERG

TTWT Rating: ★★★★★

A green pendulum bob hangs from a thin wire over a sandy surface. In the background, a large, faint footprint is visible in the sand. The entire scene is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk

Standards?

We can do better



- by James Christie

At EuroSTAR 2013 I had a brief disagreement about software testing standards with a member of the working group that is developing ISO 29119, the new standard. To be more accurate, I was one of a group of sceptics pressing him. He was putting up a battling defence of standards and made a very interesting and revealing point. He insisted that the critics of standards don't understand their true nature; they are not compulsory.

The introduction to standards makes it clear that their use is optional. They become mandatory only if someone insists that they must be applied in a certain context, often by writing them into a contract or a set of in-house development standards. Then, and only then, is it appropriate to talk about compulsion. That compulsion comes not from the standard itself, but from the contract or the managerial directive.

I found that argument unconvincing. Indeed I thought it effectively conceded the argument and amounted to no more than a plea in mitigation rather than a plausible defence.

Even a cursory analysis of this defence reveals that it is entirely specious, merely a statement of the obvious. Of course it is a choice made by people to make standards mandatory, but that choice is heavily influenced by the quite inappropriate status of IEEE 829 and, in all likelihood ISO 29119, as standards. Calling them standards gives them a prestige and authority that would be missing if they were called guidelines. The defenders of standards usually want it both ways. They refer to standards when they are making an implicit appeal to authority. They refer to the standards as guidelines when they are on the defensive. That doesn't wash. Standards and guidelines are not synonymous.

The line of defence that “other people” make standards mandatory struck me as very interesting because it was entirely consistent with what I have long believed; the rationale behind standards, and their implicit attraction, is that they can be given mandatory status by organisations and lawyers with a poor grasp of software testing.

The standards become justified by the mandatory status assigned to them by non-testers. The justification does not come from any true intrinsic value or any wisdom that they might impart to practitioners. It comes from the aura of the word “standard” and the creators of standards know that this gives them a competitive advantage.

Creating standards is a commercial activity

Standards are not produced on a disinterested “take it or leave it” basis. They do not merely offer another option to the profession. Standards are created by people from the companies who will benefit from their existence, the companies who will sell the services to implement the new standard. In my experience heavyweight, document-driven processes require large numbers of expensive consultants (though not necessarily highly skilled consultants). Creating standards is a commercial activity. The producers of standards are quite consciously creating a market for the standards.

If the creators of standards were merely expanding the market to create a profitable niche for themselves that might not be a big deal. However, the benefit that accrues to them comes at the expense of everyone else.

It comes at the expense of the testers who are frequently committed to following inappropriate and demoralising practices.

It comes at the expense of their employers who are incurring greater and unnecessary costs for results that are poorer than they need be.

It comes at the expense of the whole testing profession. The standards encourage a dangerous illusion. They feed the hunger to believe, against all the evidence, that testing and software development in general, are neat, essentially linear activities that can be rendered orderly and controllable with sufficient advance documentation. Standards feed the illusion that testing can be easier than it really is, and performed by people less skilled than are really needed.

As I said in my EuroSTAR 2013 tutorial, testing is not meant to be easy, it's meant to be valuable.

Good contracts or bad contracts?

It is understandable that the contract lawyers find standards attractive. Not only do standards offer the lawyers the illusion that they promote high quality and define the correct way for professionals to work, they also offer the lawyers something they can get their teeth into. A standard makes it easier to structure a contract if you don't know about the subject area. The standard doesn't actually have to be useful. The point is that it helps generate deliverables along the way, and it requires the testers to work in a way that is easy to monitor.

Contracts are most useful when they specify the end, or the required value; not when they dictate how teams should reach the destination. Prescriptive contracts can turn unwarranted assumptions about the means into contractually mandatory ends.

I once faced what looked like a horrendously difficult challenge. I had to set up a security management process for a large client, who wanted assurance that the process would work effectively from the very start. This had been interpreted by my employer as meaning that the client required a full-scale, realistic test, with simulated security breaches to establish whether they would be detected and how we would respond. This would have been very difficult to arrange, and extremely expensive

to carry out. Failure to deliver on the due date would have resulted in heavy weekly penalties until we could comply. However, the requirement was written into the contract so I was told we would have to do it.

I was sceptical, and went back to the client to discuss their needs in detail. It turned out that they simply needed to be reassured that the process would work, smoothly and quickly. Bringing together the right people from the client and supplier for a morning to walk through the process in detail would do just as well, at a tiny fraction of the cost. Once I had secured the client's agreement it was straightforward to have the contract changed so that it reflected where they really wanted to end up, rather than stipulating a poorly understood route to that destination.

On many other occasions I have been stuck with a contract that could not be changed and where it was mandatory for testers to comply with milestones and deliverables that had minimal relevance to the real problem, but which required such obsessive attention that they detracted from the real work.

Software testing standards encourage that sort of goal displacement; management attention is directed not at the work, but at a dubious abstract representation of the work. Their attention is directed to the map, and they lose sight of the territory.

We can do better

Sure, no-one **has** to be a sucker. No-one has to buy the snake oil of standards, but caveat emptor (let the buyer beware) is the legal fall back of the huckster. It is hardly a motto to inspire. Testers can do better than that.

What is the answer? Unfortunately blogs like this preach largely to the converted. The argument against standards is accepted within the Context Driven School. The challenge is to take that argument out into the corporations who are instinctively more comfortable, or complacent, with standards than with a more flexible and thoughtful approach.

I tried to challenge that complacency in my EuroSTAR tutorial, "Questioning auditors questioning testing". I demonstrated exactly why and how software testing standards are largely irrelevant to the needs of the worldwide Institute of Internal Auditors and also the Information Systems Audit and Control Association. I also explained how more thoughtful and effective testing, as promoted by the Context Driven School, can be consistent with the level of professionalism, accountability and evidence that auditors require.

If we can spread the message that testing can be better and cheaper than corporations might start to discourage the lawyers from writing damaging contracts. They might shy away from the consultancies offering standards driven processes.

Perhaps that will require more than blogs, articles and impassioned conference speeches. Do we need a counterpart to testing standards, an anti-standard perhaps? That would entail a clearly documented explanation of the links between good testing practices and governance models.

An alternative would have to **demonstrate** how good testing can be accountable from the perspective of auditors, rather than merely asserting it. It would also be directed not just at testers, but also at auditors to persuade them that testing is an area where they should be proactively involved, trying to force improvements. The testers who work for consultancies that profit from standards will never come on board. The auditors might.

But whatever form such an initiative might take it must not be called a standard, anything but that!



James is a self-employed testing consultant. He has 30 years of experience in IT. He has worked as a test manager, IT auditor, information security manager, project manager, business analyst and developer.

He is particularly interested in the links between testing, governance and compliance. He is a member of ISACA, the Information Systems Audit and Control Association. James spent 14 years working for a large UK insurance company and then 9 years with IBM working with large clients in the UK and Finland. He has been self-employed for the last 7 years.



We just filled
The State of Testing survey....
And YOU?



☒ Take the Survey Now



Are You Smarter than A Test Manager?

- part 1

- by Mike Lyles

In the popular US TV game show, "Are You Smarter Than a 5th Grader", we watch as adult contestants attempt to answer questions from the 1st to 5th grade level. The contestants, who have moved far beyond fifth grade, feeling very confident that they can answer the questions, however, as the show progresses, both the contestants and the audience realize that the questions are tough, and knowing the right answers is not always that easy. In fact, by the time they reach fifth grade level questions, they are usually guessing at the answers, and hoping they are getting the answer right.

This is likely the same case in testing organizations. How many of us are years beyond the entry into the testing profession, but are unsure of the perfect answer to some of the core testing questions and hot topic discussions today? Are there areas where your organization is not in alignment? The answer is probably yes. Being a testing practitioner requires understanding the many schools of testing and who is supporting them.

Over the past year, I have examined where organizations sometimes fail to remember the core fundamentals of testing, I began to focus on the central function which drives the teams: test management. I began my research by taking interviews from well-known names in the industry, test managers, discussions from Twitter, LinkedIn, and blog sites. I created a test management survey, which was taken by over 275 people, providing information around the struggles that testing organizations face, areas of improvement, and most importantly where everyone feels teams are not aligned.

This article is going to be a two-part series. Part 1 will review the interviews. Next month, we will look at Part 2, the survey data.

Let's get started with Part 1. In the past year, I have shared my research in multiple conference tutorials and have engaged the audiences in discussions around the alignment debates on many topics. What I have found is that we are not alone. We face similar problems. As you read over these results, I challenge you to compare this to your own organizations.

The Job Description

When talking with many in the field, everyone agrees that the test manager job description is not clearly defined. Therefore, I started pulling together some notes on what a typical job description would look like for a test manager. It would probably look like this:

- Involved in planning, monitoring, risk management, and control of testing projects
- Understand the overall project scope – deliver test strategy and approach
- Proficient in diplomacy, negotiation skills, conflict resolution, and professionalism
- Front line point of contact / representative for the testing team
- Blocking distractions for the testing team
- Identifying training needs and monitoring career growth of the team
- Strong understanding of SDLC at all levels and how testing is integrated
- Understanding of testing tools and methodologies
- Encourages consistencies in the processes
- Responsible for reporting status / measurement of the testing process
- Must be adaptable: CHANGE is constant in the Test Manager world

But let's face it. If we put together the 'real' job description, would it look more closely like this?

- Must be able to deliver the product on time:
 - Without final scope defined
 - With delays in development schedule
 - With limited budget and test team
 - With reduction in testing coverage in order to meet deadlines
- Must be willing to allow stakeholders and development team override you
- Must be able to handle highly demanding customers
- Must be willing to accept poor code quality
- Required to continuously justify the existence / need for testing
- Prepared to accept blame for defects in production
- High tolerance for pain due to the constant changes with no changes to the testing schedule or scope
- Comfortable with organizational belief that testing is viewed as "overhead"
- A doctorate in psychology is preferred

Yes, this list is very humorous to read. However, we all know many of these statements are true for our organizations. And, while I initially laughed out loud when I saw the note regarding a doctorate in psychology, the more I have thought about this, the more it makes sense. The testing profession requires us to understand how people think, how project teams work together, and it's a very social profession. We have to be able to deliver bad news to the people that we know are not going to enjoy hearing it. We have to be able to comfort the development teams when we share with them that they delivered a broken product. If you are unable to think about the reactions and the way that organizations are going to behave, you will surely struggle as a test professional.

I turned to some testing folks in the field, interviewing them on four questions. Let's review their responses:

QUESTION 1: What is the Test Managers Role?

Michael Bolton says "A test manager's role is to coordinate and to enable testing work—investigation of products and projects—and the reporting of it, with the goal of revealing timely quality-related information to the project communities and the business. A typical focus is on finding important problems that threaten business value, but in general, in my view, the manager's job is to provide resources, remove obstacles, and foster the development of skill such that the testers can do their best work. One aspect of this—often overlooked—is that the test manager needs at least to be a good enough tester to observe, evaluate, and guide the work appropriately. Another aspect—also often overlooked—is that management work is different from testing work, and requires its own set of training and skills."

Jerry Weinberg says "I've seen several handfuls of 'test manager' role definitions, so there's no one way to answer this question. I've seen managers without teams, teams without managers, and all variations in between. My own preference would be to reserve the title 'manager' for someone who is actually a manager, and would therefore ideally have the skills of a manager."

Michael Larson says "A Test Manager has two hats that they have to wear. The first is the people part, which means developing and working with testers so that they learn, grow and develop their talents, improve their craft, and can become more effective testers. The second part is the actual testing strategy part, where time tables, coverage, and realistic goals need to be established. We can't test everything; we can't even test a small part of everything. Priorities have to be established, and those priorities may be at the whims of things (and people) beyond our control. The Test manager needs to be able to balance those two items, and at times, help make sure that their testers are effective and doing what will help deliver value to the organization, while shielding them from the strange whims of the stakeholders and the other aspects that can make the testing process a bit, shall we say, unpleasant."

Rex Black says "A test manager needs to manage in three directions: inward, outward, and upward. Inward management refers to running the test team and its day-to-day and long-term activities. Outward management refers to managing communication of test activities and results to peer-level managers. Upward management refers to communicating test results and the value of testing to managers above the test manager. All three of these directions are important for success. In order to be successful; test managers need skills in the following areas: project management, communication, and influencing. Inward management requires project management skills, because testing is a critical and complex part of any project. Outward management requires communication skills, because test managers need to be able to deliver messages that are often not popular. Upward management requires the ability to influence, because testing is not the most popular activity in many organizations, and the value is not obvious."

Scott Barber says "I believe that what most organizations title 'test manager' is actually a 'tech lead of testers' and that the orgs that have a role that does what I believe a 'test manager' should do, generally call that role 'Director of Testing' or 'VP of Quality'. As far as I am concerned, if you do not set salaries, do annual reviews, manage budgets, etc. you are not a 'manager'. If you're going to be a manager, be a manager. Managers manage people & resources with the goal of making the business successful. If you want to focus on technology, quality, or (basically) anything other than profit, don't be a manager – be a 'lead' in your area of specialization."

Andy Tinkham says "I see the roles of a Test Manager to be that of facilitator and coordinator. I expect them to have the big picture view of the system and team actions, and ensuring that the team focuses on providing the most valuable information in the time they have. Much of the value of the information provided derives from the risks that the information addresses and the questions that the information answers. I see the Test Manager's role as discovering what risks and questions are important to the business and then ensuring that the team is used as efficiently and effectively as possible to address those risks and questions. They also need to be able to communicate the status and results of the testing team out to the rest of the team and stakeholders."

A good test manager needs to be able to talk to both the technical team members and to the business and facilitate the communication between the two groups (who don't always speak the same language). Relationship building is a critical component as well – both internally on the team and externally. Communication skills are essential.

QUESTION 2: What are the Challenges that Test Managers Face Today?

Matt Heusser says "The biggest problems I have seen in management is when the company itself does not clearly define what a manager is expected to do. This leads to all kinds of problems. In many cases, the test manager has one 'boss', but in reality has a half dozen or more stakeholders, and often they expect different things. Most of the companies I am working with lately do not see testing as a centralized function -- the testers are in integrated product teams -- so test management is more of a matrixed role, a practice manager role. In that case, the project management aspect is decreasing. So where ten years ago I thought project management was critical to be a test manager, today I would only say this on a large program where the testing is handled as a mini-project."

Jerry Weinberg says "The number one challenge is that there's no clear understanding of what a test manager is, so it's a great challenge to satisfy the 'role.' Related to this challenge is the challenge of being the scapegoat for the mistakes made by everyone else in the org."

Doug Hoffman says "The challenges are technical, communications, and political understanding. Technical challenges include understanding testing, the product, and the technical context. Communications challenges cross the organization, customers/users, management, and modes (written, verbal 1:1, verbal 1:n, online, face-to-face, etc.). Political understanding is about identifying, learning about, and communicating with stakeholders within and outside the organization."

Rex Black says "The test manager role has many challenges. It is one of the most difficult management tasks in any organization. The role does vary, but less so than many people might suspect: most of the challenges are common across industries, across countries, across cultures."

Scott Barber says "Test Managers are universally under-trained, under-powered & under-informed; have unbalanced accountability, responsibility & authority; are provided job descriptions/role responsibilities that do not actually mesh with, or compliment, the overall goals of the project, product and/or business."

There is no commonality (beyond the word 'test') in the roles & responsibilities of test managers across the industry. I guess that's not exactly true, the commonality is that because those 'above' test manager on the org chart don't get & don't want to get testing, they want someone on the org chart to either:

- just make sure it gets taken care of OR
- point the finger at when it doesn't get taken care of the way they expected."

Andy Tinkham says "The following challenges exist:

- Conflicting visions of what the software should do or how it should work across stakeholders
- Reluctance to share information because it might make someone look bad
- Intrapersonal differences
- Team member allocation to tasks
- Us vs. them cultures
- Dealing with blame (why didn't the test team catch this before release?)
- Justifying priorities"

QUESTION 3: How Important is People Management as a Test Manager's Skill?

Michael Larsen says "I would say the people part needs to be about 50%, if not more. We want to help testers become competent and effective. Therefore, it's important to give them direct and targeted feedback about what they are doing, how they can do it better, and to give them props when they do that. The worst feeling is to be given no direction, and then to only get feedback at seriously infrequent levels, and then it's just to hear 'you're doing it wrong' or 'you're not being effective'."

Doug Hoffman says "People management is probably 5% of the job, but it's the most visible and incredibly important. It's the foundation upon which the test organization succeeds or fails. To me, the rewarding part of management is the leverage - enabling all of the people I manage to do their best work and the right work. Fail at the people management and the test organization will almost certainly fail. Failing with other skills usually leads to personal failure and a weak contribution by the test group."

Rex Black says "People skills are certainly critical for test managers. Project management, communication, and influencing are important skills, and these are all people-related skills. A person without people skills is unlikely to be successful as a test manager."

Andy Tinkham says "For any manager, I see people management as a pretty critical skill, as it's the foundational element of managing. We don't manage tests, we manage testers. I'd call someone without the people management skills a test lead, not a test manager. Thus, I'd put it at somewhere around 75-80% of the foundation for success, I think."

QUESTION 4: What are the Major Conflict Zones in a Testing Organization?

Matt Heusser says "Internally I still see a lot of flak between dev and test. There's also some operational conflict, when test needs an environment up and the release team doesn't have it ready, etc. One pressure I do see lifting is the testers as quality gatekeepers vs. PM who want to ship conflict - that seems to be decreasing as we learn to talk about risk in more objective terms."

Michael Bolton says "Testing work is about revealing information about the product. Sometimes that information is painful, and some people are uncomfortable with hearing it. For that reason, as Jerry Weinberg suggests, the foundational emotional position for a tester and a test manager should be empathy. Jerry further points out that, decision about quality are always political and emotional, so in addition to emotional congruence, a test manager needs political acumen. Some testers and test managers believe that they are guardians or gatekeepers of quality. I don't think this is a helpful perspective. Shipping a quality product is a goal shared by everyone on the project. (If there are disagreements about what constitutes a quality product, the disagreement is rarely about conclusions; it's usually about premises, founded in different values and different reward systems for the parties involved.) Test managers who see themselves as quality gatekeepers will often be party to various kinds of unhappiness, until they recognize that testing is not quality assurance; testing informs quality assurance."

Jerry Weinberg says "The major conflict is with people who believe that without any testers, there wouldn't be any errors."

Michael Larsen says "Smaller teams are going to suffer from a lack of oversight and low levels of communication. Larger organizations will suffer from bureaucracy and inefficient processes. In between teams will be squeezed to do as much as possible with as few resources as they can. In short, testing is the easiest target, since they can be blamed when things slow down, as well as when things are discovered in the field. The Test Manager can provide an invaluable service to their team at this point in that they can set the standard and the expectation of what is doable, what testers can actually provide, and what decisions are made by which groups based on the information we provide... and making sure that those who get that information are held accountable for their decisions."

Doug Hoffman says "Conflict zones are generally not necessary. Testing is not inherently in conflict with any stakeholders. When preexisting conflict exists it can usually be overcome by directly addressing it to remove or mitigate the cause. It takes understanding the common ground and emphasizing it (e.g., everyone usually wants the best quality, lowest cost, in the fastest time). I find that showing respect for the value of their role (however high or low I may think the value is) almost always opens people up. My trying to understand what they need and how I can help is another way of defusing/avoiding conflict. E.g., I ask developers 'In general what information would be useful to you when I find something I think is a bug?' and then I follow up with 'How would I get that information for you?' The end result often is developers providing hooks and/or code that dumps the information, making both of our jobs easier and more effective. I get development resources to do a better job of testing without conflict."

Rex Black says "If a test manager has 'conflict zones', the test manager is already failing. The successful test manager knows his or her stakeholders, their needs, and how testing relates to their operations. The successful test manager focuses on working collaboratively with those stakeholders."

Scott Barber says "The conflict zones are anyplace zones are defined. The biggest mistake that business has made is dividing up titles, roles & phases in software development. Outside of software, where do we hear of people with a title that includes 'test'?"

- Test Pilot
- Taste Tester
- (Medical) Test Technician

- (Academic) Test Administrator
- Crash Test Dummy

And yet people test pretty much every product that is even considered for possible sale. So who does that testing? Everyone .The team. The engineers. Are some engineers especially skilled in testing? Yes. Are they known as “testers”? No. Do they report to a different manager? No. Do they sit in a different room? No. The testing simply happens in-line, continuously and collaboratively. Problems are fixed or listed to be dealt with later. So, the simple fact that testers think of themselves as, or are seen as, separate in any way from development is the heart of the problem.”

Andy Tinkham says “Every team is different – I think that there is the potential for conflict between any two groups, and it all comes down to the relationships built and the trust that exists between the groups. An organization that has a culture of everyone working together towards one end (delivering a useful software product that meets the needs of the software’s target audience) might have very little conflict. Organizations that have distrust between groups are going to see the most conflict where there is the most distrust. Any group that thinks another group isn’t holding up their end of the process, not doing their tasks correctly, or not working as hard as another group may well have conflict. Any group that feels another group just makes demands without understanding the impacts of those demands is going to likely have conflict. It really comes down to the areas that lack communication and/or trust. Of the groups you list, it probably most likely manifests with PMO, Development and Release Management, but could be all of them.”

What does Part 1 of this series tell us?

While there are many diverse thoughts around test management, many of the core beliefs are quite similar. There is agreement on things such as: the role is not clearly defined, test managers should be managing (or else they are test leads), the role requires strong communication and social skills, and most importantly, the age-old conflict between testing and other organizations exists everywhere.

I began my research to identify the gaps, but instead realized the importance of continuously sharing our experiences with others in the testing profession. Your organization has the potential to influence quality throughout the organization, and by learning from others, we can develop the best processes for obtaining support from other teams.

Join me next month for Part 2, where I will share the results of the test manager survey, as well as the hundreds of responses provided by the testing community on: why they chose to be a test manager, what things they would do differently, and most importantly, areas where they feel their organizations are not aligned.



Mike Lyles is a Sr. QA Manager with 20+ years of IT experience, working in various roles over the years. His current role is over Performance and Automation testing for all business communities within his organization. Mike enjoys teaching others in the testing profession. He has spoken at multiple conferences on test management topics, and has written multiple published articles.

You can learn more about Mike at <http://about.me/mikelyles> or on his website at www.MikeWLyles.com.

The Assumption Bias & Testing: How Does It Influence You?



Assumptions

- by Guy Mason

When testing a product, I aim to provide what I believe to be appropriate test coverage; so that I can perform the kinds of testing that will cover all of the areas that I believe are of value to the business.

There have been times in my own testing where I have discovered only later in the piece additional areas to cover, areas that I had not originally thought to scope in due to assumptions of my own. Whilst grateful to have picked up on such things before the product ships, there always remained the risk that should there be a failure to do so, or should there be issues which arise around it later in the piece, this may impact the ability to deliver the product on time and to the expected specification.

Contrary to what some might assume, this is not a product of inexperience, but is instead a product of becoming so familiar with something you are testing, often based on extensive experience with testing similar things (and where the tester holds an extensive level of domain knowledge), that this increased level of confidence can impact our perspectives with the testing that we perform.

We can try and shape our perspectives by taking a focus on the bigger picture through asking questions, questions that guide the tester as to what knowledge about the product may offer the greatest value for the business. We can then use this information to help guide us with what areas to focus on when we are testing too.

This however still does not eliminate these assumptions we hold when performing this task. This is because it too remains an externally facing exercise, as we tend to not include ourselves as completely within the equation when performing such analysis.

The issues that stem from this are comparable to something that is labeled tacit knowledge. Tacit knowledge represents knowledge that is shared on a social level, but has not yet been documented so as to exist, at least on some level, in an explicit form.

Like tacit knowledge there exists an undocumented aspect, an aspect that can equally be influenced by the social, but in this case is much more centric to the individual. In these circumstances it relates to the absence of an evaluation of the biases and the assumptions that we bring to the table when performing test design and evaluating what we feel to be relevant test coverage.

If we take the time to first analyse and document these biases and assumptions before launching in and evaluating what testing we are looking to perform, we can use this knowledge to help shape our testing, so that what is and is not covered is no longer as influenced by such factors.

Such information gives us an opportunity to identify additional areas where test coverage might have otherwise been missed, and it becomes an additional source that we can utilise for future test planning too. In addition it serves as an opportunity to gain a greater awareness and understanding of these influences that we hold too.

Taking this very human element and being mindful of it and its influences, when creating test plans, can assist with giving us greater confidence that the testing we perform will be less likely to fall short due to such influences. This then helps us achieve the kind of coverage that can better assist with the delivery of a quality product.

Guy has been involved with technology since a very young age. Throughout his lifetime he has created and run web sites, written for and assisted with the running of computer publications, been a hobbyist programmer and assisted with running computer related events. As a tester he started in his first testing role back in 2000 and has since worked across many of the leading Digital Agencies throughout London and assisted several of them with formulating test strategies to use in their business, in addition to also holding various other roles in Digital Media and Software as a Service companies.

Guy's love of technology is married with his love of testing and he has a great passion for assisting with quality improvements for projects he works on where possible. You can contact him on Twitter (@TestingQA) or via his blog (testingqablog.blogspot.co.uk).



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.

A photograph of a classroom scene where several students are raising their hands. The students are seen from behind, wearing light blue, red, orange, and green shirts. They are in front of a dark chalkboard. The image is framed by a thick black border.

In the school of Testing

for your better learning & sharing experience



Preparing for Automation: Selecting the Tools

a series by Jim Holmes

In my last article I laid out a number of factors I've learned are critical to address before starting your user interface test automation efforts. (Or address them as quickly as possible once you've started.) You may have noticed that article likely left you with more questions than answers. That's OK, because the process of getting any good testing activity in place *should* leave you with lots of questions! By the way, this article will likely leave you with even more questions—as will likely every future article...

Selecting the right toolset for your automation¹ is a critical part of your automation project's success. Choosing an ill-fitting toolset doesn't guarantee failure; however, you'll spend an extraordinarily frustrating amount of time on the mechanics of creating automation: learning the tools, writing tests, running tests, fixing broken tests, adapting tests to system changes, etc. All of that takes you away from your primary value as testers: Actually *testing* the system!

Getting the toolset right, or quickly changing away from an ill-fitting toolset, means you'll spend less time struggling with the mechanics. Your team will be better integrated with your testers, your stakeholders will be getting better information, and your testers will have time to test versus chasing broken tests.

¹I use "automation" in this series to mean "User Interface Automated Testing." I'll be explicit if I mean some other form of automation.

Goals for Automation

In last month's column I asked "Why do UI Automation in the First Place?"

It's a perfectly valid question, and one your team needs to answer. As part of answering that question, get distinct goals in mind for your automation effort. Goals for your effort might include

1. Increasing communication about testing across your team
2. Helping cut down the number of regression bugs that escape to your customers
3. Helping your team build what your stakeholders actually need
4. Freeing your testers from rote regression tests so they can do more important testing

Note I never once mentioned metrics like "ensure 100% test coverage," "create 5,000 automated test cases," or "convert 95% of manual tests to automated." Those are incredibly awful, horrible, nasty, smelly metrics that in my view have no place in any project. They're extremely shallow metrics that don't indicate the value of tests. Make your goals something that's more effective in helping your team deliver true value to your customers.

A Tangent: Nomenclature

Before we head off much further into the depths of selecting tools, let's take a moment to lay out some terms that will help us keep clear the various moving parts involved in automating web browser tests.

At the lowest level, naturally there's the *browser* itself. The browser navigates to web resources, loads and renders content, executes JavaScript, etc.

The next level up is the *driver*. This is an automation tool such as Selenium WebDriver, Microsoft's CodedUI, or Telerik's Testing Framework. The driver is responsible for communicating commands to the browser (navigate, load, etc.) and inspecting the page inside the browser. Generally speaking drivers don't actually execute the automation tests themselves. Additionally, most drivers don't even know how to do actual checks or comparisons ("Is this heading block correct?" "Is the button visible on the page?") Drivers simply pass instructions to the browser and pull back bits of information from the browser. Frameworks reside on top of the driver to handle these functions.

A testing *framework* fills the test execution and comparison/checking gaps. Frameworks such as JUnit, NUnit, or MSTest provide containers which driver code can handle the browser operations. Testing frameworks also provide comparison and assertion methods for validating conditions on the page.

Specification or language frameworks enable teams to write specifications in a more natural grammar. Drivers and testing frameworks don't communicate intent particularly well, especially to analysts or stakeholders. Frameworks like Cucumber, Saunter, Fit/Fitness, and Robot offer up ways to express intent using grammar similar to "As a registered customer, I want to add bacon to my cart so I can purchase it."

Commercial tools such as HP's QuickTest Pro, Microsoft's Visual Studio tools, or Telerik's Test Studio often abstract away some of the direct browser interaction at the low-level driver. Good tools in this vein still provide you direct access to the driver and browser via unfettered control at the driver or code level.

Who Will Use It?

With nomenclature out of the way, clearly understanding the users of your testing toolset is critical when selecting the tools.

If you're looking to involve your stakeholders in creating specifications that tie directly to tests, then you'll want to look to a tool that gives you something akin to plain language. Cucumber, Fitness, and Robot are examples of specification-style frameworks that fill these needs.

Teams that will rely more on developers and coding testers to get their tests created may be more interested in staying straight at the driver level (with a test execution framework), or perhaps a BDD-style framework such as SpecFlow or Saunter.py.

What's the Environment for the Tests?

Your system's environments play a huge role in deciding what toolset you'll select.

Are you planning on running your tests only at your developers' or testers' systems? (Please say "No!") If not, here are some things to take in to consideration:

1. Do you have multiple environments for development, testing, pre-production, and production?
2. Which environments will you be targeting for testing?
3. Who owns deployment to those environments? Do they require a manual push process, or can you tie your automated tests to some form of automated build process?
4. Are you able to deploy tests in an open source tool to your pre-production and production environments?
5. What's the workflow for getting things like execution agents and platform requirements deployed to those environments?

Many larger corporations have multiple environments, each with different deployment requirements—especially when those corporations are under regulatory compliance in the financial or healthcare domains, for example. If you're working in these environments you'll likely need a toolset that support easy configuration for things like base URLs, database connection strings, file system resources, etc. Note that by "easy configuration" I really mean "the ability to write custom configuration libraries in as simple a fashion" because few toolsets have out-of-the-box support for complex environments!

What are the Current Technologies in Use?

Choosing an automation tool isn't wholly dependent on the technologies your team's using to develop your system; however, it often has an impact. For example, if you're working on a Windows Forms project, you'll have fewer options than if you're helping test a Ruby on Rails web application. There are a number of technology factors you'll need to understand about your environment:

- What is the basic architecture of your system? Thick client? Standalone desktop application? Web application with mobile support?
- What language is your system being developed in?
- What devices will your users access your system through?
- What components are running in your application's architecture? (Web server, business layer, database server, distributed content network servers, etc.)

Good automated UI tests don't rely just on the user interface; they use test oracles to probe the underlying system to ensure databases are properly updated or system configurations are altered. This means you'll need to understand how to interact with your low-level system's APIs. Are there appropriate web service endpoints you're able to access? What drivers are available for your database if you need to create direct connections?

Just because a system is written on one platform doesn't mean you're restricted to the same language or platform for your tests! This is particularly the case with web applications. Use Telerik's Test Studio to cover Rails applications. Use Adam Goucher's Saunter.py Python framework to automate tests for .NET applications. Use Jeff Morgan's Ruby Pages gem to drive out functionality on Java applications. Even in the WinForms domain you can look to Ruby code that will exercise your Windows desktop applications.

While the technology stack does put some constraints on your automation approach, more critical is your team's makeup and skills.

What's the Team Structure?

How your team is structured has a tremendous impact on your automation efforts. I'm a firm believer the best chance for success in any automation project requires involvement from the entire team: stakeholders, designers, developers, business analysts, and of course testers.

If your team suffers with a much more constrictive, stovepiped structure then you'll want to avoid tools that encourage more collaboration across those boundaries. Instead, you'll want to focus on tools that will boost your testers' productivity, perhaps at the cost of some overall flexibility.

Perhaps your developers can't or won't assist in actually writing the tests, but they may be available for writing backing infrastructure to handle setup, teardown, and configuration. They may also be able to write appropriate test oracles for your testers. If this is the case, then your testers will be able to take advantage of just about any toolset they're comfortable with. Being able to rely on developers for the "heavy lifting" of writing calls to web services or the database can let the testers focus their main skills of building great tests.

What are the Team's Skills?

You've got to allow for your team's skills when selecting an automation toolset—and not just the current skills, but those that can be learned while adopting a new toolset.

Are you choosing test tools for a developer-centric group comfortable with .NET? Will you have time to learn a new language like Python or Ruby in order to use an interesting toolset like Ruby Page Objects or Python's Saunter.py?

Has your team had previous automation experience in the technology stack you're currently working in? Do they understand things like the page Domain Object Model and asynchronous operations? Does your team understand how to build well-designed software—and that those same design principles are critical to well-crafted test suites, especially if you're using a completely coded approach like WebDriver or Watir.

How Long Will You Need Your Automation?

It's not enough to just consider your team's skill as you're building up your automated suite. You have to take into account how long your system will need the automation you're creating. If you're relying on a smaller open source project, will it be around in three or four years, or is it likely you'll need to change to a new toolset? If you're looking to a commercial tool, will you have budget in the out years to continue getting support?

You also need to consider the time it will take to onboard new team members. Will they have to learn new development languages in addition to getting up to speed on all the other testing activities in your organization?

Try Things!

Once you've worked through the long list of questions I posited above (you've likely noticed I gave few answers!), you'll need to put in time exercising a few of your top options for toolsets. Part of your selection process absolutely needs to include time on the schedule doing real work with the

tools to see how they “work in anger².” An extended prototype period or proof of concept is the only way to get a good feel for how the tool will work for your teams.

Get time on your schedule for working through your tools, and make sure it’s *adequate* time for a serious evaluation. Ensure you get everyone, *EVERYONE*, on the team involved in the trials. You need to understand which of your team will have difficulties with what tools, and which of them loves the other tools.

Selecting a tool is a significant effort. Your team needs to make sure they’re comfortable as a group that the chosen tool will help bring value to your software efforts, not just turn into yet another technology time sink.

Keep in mind the goals for your automation that were discussed near the start of this article. Every one of your selection criteria ought to align with those goals. Otherwise your team’s in for a long, frustrating experience!

A Smattering of Tools to Consider

Here are a few tools you may consider during your evaluation. The list is far from complete, so don’t rely on my article for your single source of potential toolsets!

Name	Platform	Notes
Drivers		
WebDriver	Bindings for all popular languages.	Web standard for automation. Extremely broad use, which means lots of blog posts and examples. Supports all major browsers and some mobile devices.
Watir & FireWatir	Ruby	Driver for Internet Explorer and Firefox.
Microsoft Coded UI	.NET	Nicely integrated into Visual Studio. Supports IE only. Handles Silverlight and WPF.
Telerik Testing Framework	.NET	Supports cross-browser testing, Silverlight, and Windows Presentation

²I heard this phrase from a customer evaluating our tools. It was his way of saying “in the real world” and I loved his twist on that phrase.

		Foundation desktop applications.
Frameworks		
Saunter.py	Python	Targeted to simplify automation through ease of use in Python.
Cucumber	Ruby	Not bound to any one driver. Gives a grammar-based approach for writing specifications.
Fit/Fitness	Several popular languages	Table and wiki approach for grammar-based specifications.
Robot	Java	Framework on top of WebDriver. Abstracts out the hard parts of WebDriver, works best when web pages follow conventions around naming and ID values.
Other		
Sikuli	Python	Graphical automation. Allows more complex scripting via Python.
Commercial		
QuickTest Professional	Several supported.	HP's large automation tool. Extensible to support various platforms like Flash, SAP, etc.
Telerik Test Studio	.NET for drivers and code	Web, Silverlight, and WPF automation. Also supports load and performance.
Microsoft Visual Studio	.NET	Web, Silverlight, and WPF automation. Also supports load and performance.

Jim Holmes is the Director of Engineering for Test Studio at Telerik. He has over 25 years in the IT field in positions including PC technician, WAN manager, customer relations manager, developer, and yes, tester. Jim has held jobs in the US Air Force, DOD sector, the software consulting domain, and commercial software product sectors. He's been a long-time advocate of test automation and has delivered software on a wide range of platforms.

He co-authored the book Windows Developer Power Tools and blogs frequently at <http://FrazzledDad.com>. Jim is also the President of the Board of Directors for the CodeMash conference held in the middle of winter at an indoor waterpark in Sandusky, Ohio.





Rants & Ramblings of a Mobile Tester

- by JeanAnn Harrison

When to and When not to Automate Your Mobile Testing

Have any of you asked this question or some simile of this question, "what is the best tool for mobile testing?" Every day, I am asked by someone, from somewhere in the world, this very question. I've done articles, I've done webinars, I've spoken at conferences on various mobile testing subjects which are not even related to automation, yet "there's always someone in the crowd" asking this question. And the answer is...

Are you ready? Do you REALLY want to know the answer?

How about you read this full article before I answer this question? I challenge you NOT to cheat and skip down to the end. First, let me give you some other fantastic tips you can implement right away? Deal?

There are some fantastic commercial tools on the market today. I remember when I first got into the mobile testing business; there was nothing for mobile, at least none that were more universally applicable to testing. Today we have such a level of complexity of devices between the configurations, various operating systems and browsers used for mobile, the types of mobile applications, the complexity of those applications and the interactions of mobile applications with other mobile applications. The mobile testing project can be overwhelming. Let's just throw in user experience testing, performance and hardware inter-dependency testing into the fold as well. Yikes! What's a tester to do?

So when I speak of “mobile testing”, I am including all of the above listed types of testing. However, most testers, especially those new to mobile testing speak of functional testing only and regard functional testing as thorough test coverage. However, many who have read my articles listened to me at a conference or during a webinar, have heard me speak about “testing beyond the GUI”. In today’s mobile testing projects, testers run up against a demand for quick, short testing runs. Thus, testers are told “automate everything”. Well, I wish I could say, “Yes, let’s do that” but it’s not possible. You can’t do it in desktop/web applications, what makes anyone think automating everything for mobile applications is possible to achieve thorough test coverage? If anyone disagrees with me on this point, I challenge you to prove otherwise. I am open to learning something new, but you must prove it.

With the complexity of mobile testing, we do need to find more ways to automate so we can spend more time learning through Exploratory Testing methods. Here are some concepts to consider when applying automation to a mobile testing project:

1. Planning out your tests which include defining the types of tests required for solid test coverage
2. Usability & Configuration Test Comparisons
3. Test Coverage
4. Speed of Test
5. Maintainability
6. Regression & Functional Testing
7. Testing Beyond the GUI
8. One Size Does Not Fit All

Concept 1: Planning out tests which include defining the types of tests required for solid test coverage.



In mobile testing projects, so little time is given for planning out the tests. More often than not, people test just the general functionality, perhaps some user experience tests and call the task complete. However, not all applications desktop/web/mobile work the same. Mobile apps are classified into 3 different categories: mobile web/website, mobile hybrid and mobile native apps. So is there really a need for performing a traditionally defined load test on a mobile native application like you would do so for a website? What

would it matter how many people are using the mobile native app at the same time? The load isn’t hitting one machine/processor/instance.

So before testing begins, an assessment of what kinds of tests should be done for the project. Once the list of the types of tests are complete, the team can then draw some conclusions about which tests can be automated and which tests cannot be automated.

Types of tests to consider are: functional, regression, usability, performance, stress, load, system integration, trainability, user experience and configuration tests.

Now with mobile software, the definition of performance testing as previously alluded to earlier is different depending on the type of mobile software. Are you testing a mobile website or mobile web app? If so, the code is usually HTML, Javascript, CSS etc. and the code is sitting on and utilizing a webserver which the mobile device communicates. There is no code downloaded onto the mobile device, therefore, the mobile device testing dramatically decreases because most of the testing would lie on the server side. Conversely with native apps, the definition of a Load Test might refer to how many native apps can be in use while utilizing the application under test as you can have more than one native app open on a device. However, this isn't a critical test, so the question is, do you consider this test important for ensuring your application works as expected from the user perspective?

Prioritizing your test types based on what is most critical to the user experience is what can help to organize your mobile testing, define which tests should be automated and which should not. You may wish to consider creating a mind-map. Check out James Bach & Karen Johnson's mind map "Getting Started with Mobile Testing" as way to start consider planning your own mobile testing project. <http://www.flickr.com/photos/softwaretestingclub/7159412943/sizes/o/in/photostream/>

Concept 2: User Experience Testing & Configuration Test Comparisons



The mobile software user is one who has a very short attention span as Dr Philip Lew, CEO of XBoSoft, Inc. reminds us in his April 2013 STPCon session called Evaluating and Improving Mobile App User Experience. User Experience testing is the big umbrella of user testing which can include both user interaction and non-user interaction and has an effect on how the user feels about using the mobile app. Usability testing is human interaction with the mobile app and often times, only usability testing is conducted for mobile.

Do you compare what your mobile app looks like and behaves on various devices? Some types of comparisons might include what objects appear on the tablet version versus the phone version, does the screen rotate on both devices displaying the same objects, are the functional icons displayed the same from one device type to another, does the content appear to fill out the screen on a tablet and mobile phone, does the content fill out the viewing screen once the device is rotated 90 degrees? With configuration comparison testing, the tester should work with the project stakeholders to find out which tests have a high value on the user experience. Question is how much of these tests can you automate? However, the BIG question is not the "how" but the "why" or should you automate the test? Further considerations discussed can help you answer this BIG question, but be sure to think about these tests in your planning.

Concept 3: Test Coverage

Not all test cases will have one script written to complete the test. You might write a brilliant script to test the objects appearing on one screen for the Android Phone but will this script be applicable on the Android tablet or the iPad or iPhone? Probably not. Does it make sense to write 2 automated scripts or 4 automated scripts based on the device? If you're testing a proprietary mobile device, like a medical heart monitor or a point of sale/ordering device used in a restaurant, then perhaps writing automated scripts for much of the functional, user interaction behavior is efficient. Maybe not.



The tester and stakeholders should be asking "does it make sense to apply automation here" which is why Concept #1 is vital before calling test design complete and the decision to automate all or most of your testing. And re-evaluating your test coverage can change with each build, so is it efficient to re-evaluate written automation scripts due to changes in developed design? Mobile software development is so volatile; developers do not have solid requirements beyond the functional GUI. How are testers able to have sufficient test coverage when really what occurs within a mobile project is Exploration? Stakeholders do not think system integration and developers design what they know. If the mobile app produces corrupt data

because the mobile app didn't account for too hot temp while charging and doing a database search prior to sending that data to another source, then a requirement should probably be developed. Unfortunately, most project teams aren't aware to even consider such a situation; therefore, the problem shows up in production.

Builds become volatile due to lack of knowledge and experience. We don't know what we don't know. We need to learn, we need to explore more. As we learn, builds rapidly change within a project. Creating automation scripts within a certain project without considering changes can be waste of time or worse, give incorrect test coverage assurance.

To be continued in next issue...



Jean Ann has been in the Software Testing and Quality Assurance field for over 14 years including 6 years working within a Regulatory Environment and 7 years performing mobile software testing. Her niche is system integration testing with focus multi-tiered system environments involving client/server, web application, and standalone software applications. Mobile software testing includes mobile native apps, mobile hybrid apps, mobile web applications and mobile websites.


Jean Ann is a consistent speaker at many software testing conferences, a Weekend Testing Americas facilitator as well as making guest appearances. She is always looking to gain inspiration from fellow testers throughout the software testing community and continues to combine her practical experiences with interacting on software quality and testing forums, attending training classes and remaining active on social media sites.



A man in a grey suit and blue tie is drinking from a white mug. He has a surprised or tired expression. To his left is a large, messy stack of yellowed papers. In the center of the image is a large, oversized foot wearing a black sock with colorful horizontal stripes (blue, green, yellow, orange).

Taking
a break?

a click here
will take you there



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

Think better using "Descriptive-Prescriptive" approach

Testing is interesting as it is unbounded. Customer expectations constantly expand, overall development effort/time is expected to shrink and quality constantly increases!

This requires good (problem) analysis and (solution) synthesis skills. What does it take to analyse a problem and to synthesise a solution? Let us think differently...

The prerequisite to good problem analysis is a clear and deep understanding of the problem. Now how do we understand something well? Remember when you were young, you were told stories to help you understand good/evil, right/wrong etc. Story telling aids understanding. Story telling is describing something in an interesting fashion, of connecting various elements in an engaging manner, with a human touch. And that is possibly why we relate to a story better rather than mere facts enabling good understanding and fostering interesting questions to deepen the same. Describing the elements and connecting these enables us to come up with interesting questions and attempting to answer these helps us to understand better. This is what I term as 'descriptive approach'.

Now let us shift to how we possibly discover solutions to problems. Now what is a solution? A set of rules to follow, to solve. A 'prescription' that we can follow. Synthesising a solution requires us to understand the various conditions that relate to the problem and therefore a combination of these

conditions is the 'prescription' to follow. Focusing on a 'prescriptive approach' enables us to extract conditions and formulate the solution. The synthesised solution is expected to be clear, unambiguous, precise i.e. 'objective' while on the other hand problem analysis is aided by a descriptive approach which is 'subjective'.

Now let us connect these... Problem analysis requires a 'descriptive approach' while Solution synthesis requires a 'prescriptive approach'. The former is subjective and interesting involving story-telling, while the latter is cold, objective consisting of the conditions connected to form 'prescriptions' to follow to solve the problem.

Where are going with this? How does this relate to testing? Hmm. Let us look three parts in the test lifecycle...

Let us look at how we understand a product/application and formulate a good baseline for testing. We attempt to understand a system by reading the specification or 'playing' with a prior version of the system. The act of writing 'user stories' is to describe as what the system should do as a simple story. A typical specification could be dry and boring, the trick is to attach a human touch i.e. who (person or another system). The key aspects that we connect are who uses this, why do they want it, what do they value, when do they use it, how frequently do they use it, how do they intend use it. Keeping the story telling angle, and starting from the 'who' and looking at the various elements allows us to describe the system and therefore come up with questions. So when you want to understand 'storify'!

A prescriptive approach to solving the problem of understanding a system could be to see the system as collection of specific information elements that need to be connected. And the process of connecting the various elements that relate who, what, when, how, why. Think of this a small mind map to describe the system or part thereof, where we connect various elements like: various type of users, when and how much they use, how much they value this, attributes expected, environment that it needs to be supported, the state of development (new, modified), conditions that govern the behaviour, the list of features/requirements that it is made up of, the deployment environment. The process of connecting these various elements using a mind map like approach is a rational way to decompose and understand, a 'prescriptive approach'. This is embodied as a technique called Landscaping in Hypothesis Based Testing (HBT).

Ultimately stating the baseline to test as list of features/requirements/flows with attributes to satisfy is a clean 'prescriptive way' to state 'what to test' and 'test for what'.

Let us move on to test design....my earlier article (TTWT Sep 2012: "Behave yourself - Descriptive to Prescriptive") described how this is applied in test design. The design problem is one of extracting the various conditions that govern the intended behaviour. Describing how it works/intended-to-work allows to understand and identify the conditions - 'descriptive approach'. Once we identify the conditions the solution of designing test scenarios is of 'stringing the conditions' to result in scenarios to evaluate i.e. prescription. So problem of design requires 'story-fication' of behaviour conditions, a descriptive approach, while the solution to designing scenarios requires prescribing the behavioural model as a combination of conditions. Note that this is applicable to design of functional or non-functional test scenarios.

Now lastly let us discuss how this thinking approach can be applied to reporting and management. When we report progress, quality or delivery risk, it is common to report using metaphors like charts, metrics. When somebody reports dry numbers, charts, my instant reaction is "what does this mean?" and "is this good or bad?" If the metaphors i.e. charts/metrics are crisply described, then we could relate to it better and understand the status clearly. Once the status is understood well, I can compare with limits/benchmarks ('prescriptive') to formulate action plans to resolve problems.

So think better using 'descriptive approach' to analyse a problem and apply a 'prescriptive approach' to synthesise the solution. This approach forms the backbone of HBT with set of 'prescriptive' techniques and syntax of writing to aid better 'description' to enable good understanding.

Describing is very warm and human, while prescribing is a cold and machine-like! So the next time you encounter you a problem, apply the descriptive and prescriptive approach. May the heat subside!

This is the wonderful season of the year when all the problems of the year fade into a hope for a great new year. Have a lovely December. Merry Christmas and Happy Holidays!



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



[Back To Index](#)



**We wish Happy
Christmas to all
Readers!!!**

Testing PUZZLES

by Sebi



Claim your **Smart Tester of The Month** Award. Send us your answer for Puzzle b4 25th December 2013 & grab your Title.

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

Tips for Bug bounty

Bug hunting means to play. You got to maintain a childish-mature approach to keep trying new things. Searching for XSS can be a boring job if you just want the reward. You got to find a way for self-motivation, curiosity and ingenuity.

It's good to alternate tools, areas, and companies etc. to change the focus. If you found a bug type in some website try to find a similar one in another bug bounty program. Learn from others, see their blog posts.

At the end of the day there is not a template on how to find bugs in bug bounty programs. You need to constantly reinvent yourself.

[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



1			2		3		
					T		
4			5	6		7	8
9					10		
11					12		
			13				
		14					

Horizontal:

- The new tool introduced by TestPlant (8)
- Continuously raising an input signal until the system breaks down, in short form (2)
- Testing where the system is subjected to large volumes of data, in short form (2)
- Testing done on the application where bugs are purposely added to it, in short form (2)
- The first word in "object oriented programming" (5)
- It is a mature, business-ready tool for automation of web application testing (4)
- It is an open source web automation testing tool which uses Watir as the library to drive web pages (3)
- It is a tool to enhanced wiki and issue tracking system for software development projects, the first and last word in the tool name (2)
- It is a free open source tool for testing performance and scalability of web services (5)

Vertical:

- A human action that produces an incorrect result (5)
- Which company named a Market Leader in Automated Testing by VDC Research (8)
- Testing performed by the end user on software in its normal operating environment, in short form (2)
- It is a commonly used term for a specific test, in short form (2)
- It is an open source test management tool (6)
- It is a PHP/SQL based test case management (TCM) and text execution reporting system, in short form (3)
- Running a system at high load for a prolonged period of time, in short form (2)
- The first two words of "CLIF" is a load injection frame work (2)



Every Tester

who reads Tea-time with Testers,

**Recommends it to friends and
colleagues .**

What About You ?

in ne>xt issue

articles by -

Jerry Weinberg

T Ashok

Jim Holmes

Bernice Ruhland

Mike Lyles

JeanAnn Harrison

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

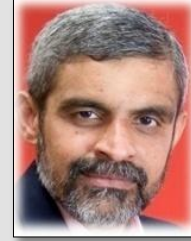
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Core Team:

Dr.Meeta Prakash (Bangalore, India)



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 kadachina |
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

