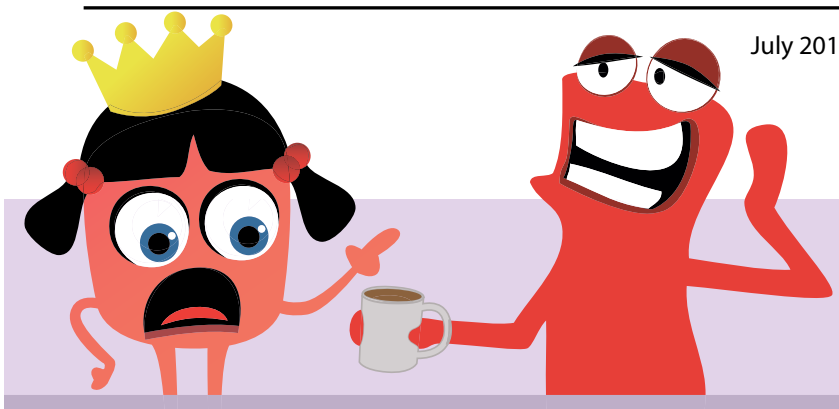
THE TESTING PLANET

July 2010 | www.softwaretestingclub.com | No: 2 £5



Yes, It's Personal

How to build healthy relationships with different job functions and team members **pages 10-11**

An Introduction to Test Automation Design

BY LISA CRISPIN

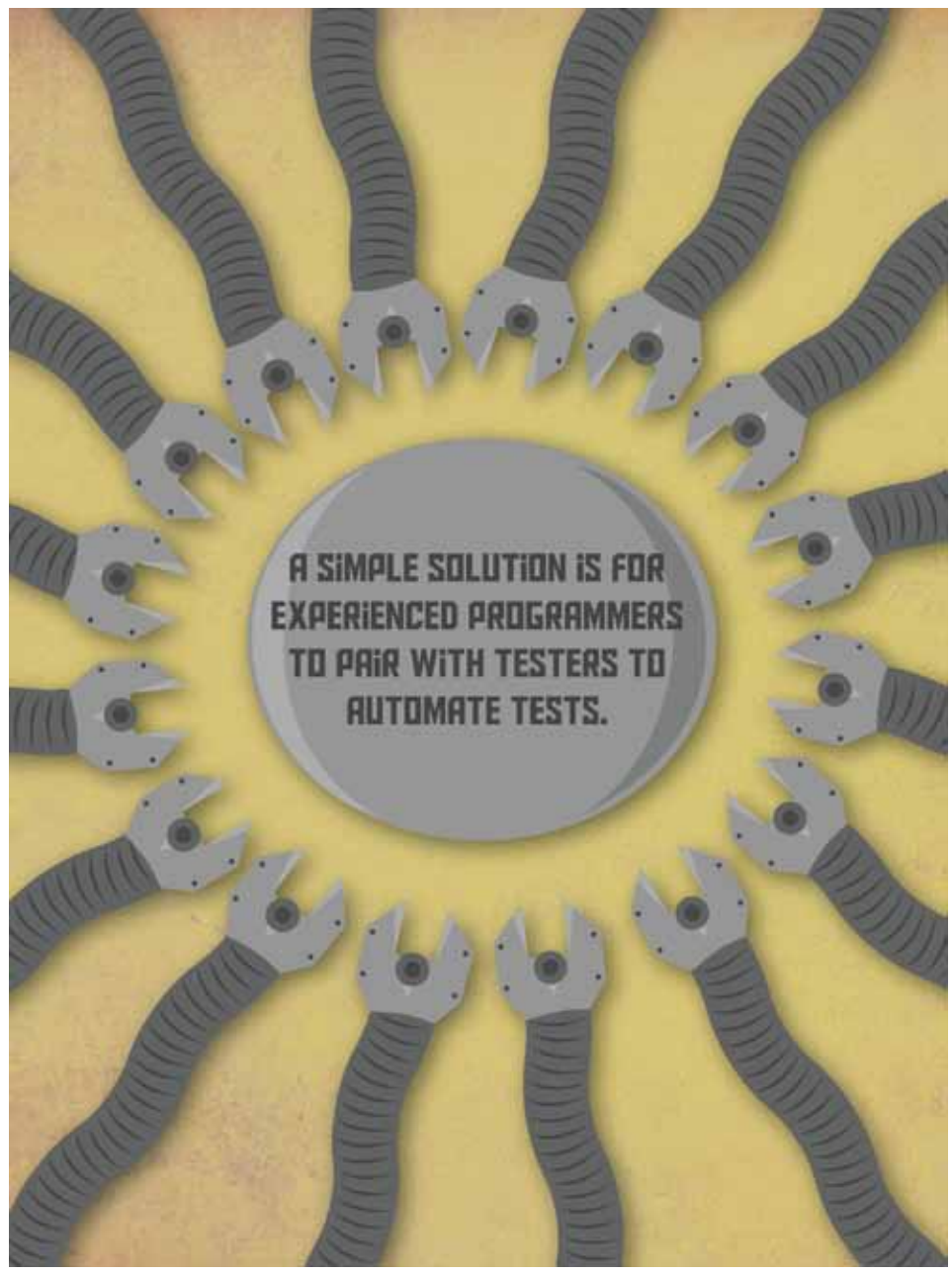
There's no shortage of excellent test automation tools available today. Many are open source, so the up-front cost is low. Some come with script capture tools or mini-IDEs that speed the learning curve. Many have excellent user guides, tutorials and screencasts available to help testers and teams learn them.

The Problem

One would think that with all this tools and help available, test automation efforts would be more likely to succeed than in years past. However, what I see all too often are teams who had no trouble creating thousands of automated regression tests – the problem comes a few months down the road, when they spend most of their time maintaining the automated tests. These testers are left with very little time with critical testing activities such as exploratory testing or collaborating with customers up-front to get examples of desired software behavior. Now they're between a rock and a hard place: regression testing manually takes too long, and so does keeping the automated tests up to date as the production code changes.

In my experience, this quandary is usually the result of poor automated test design. Automated test code is, well, code. It requires the same thoughtful design as production code. By applying coding principles such as "Don't Repeat Yourself", anytime we need to update a test to accommodate production code changes, we only have to change one module, class, include or macro, perhaps only a variable value.

Unfortunately, test automation is often done by testers who don't have much, if any, programming experience. They may be capable of capturing scripts with a tool and replaying them, but this is only a way to learn a tool, not a way to design real tests. The programmers on their projects may feel their job is to write production code, not help with functional or GUI test automation. So, the testers do their best, and



Unfortunately, test automation is often done by testers who don't have much, if any, programming experience.

IN THE NEWS

Daily Testing Tips
Tuesday has become one of my favourite days of the week because it's when testers from around the globe start posting their valuable and insightful software testing tips on twitter.
Continued Page 6

Weekend Testing fun
Weekend Testing provides the opportunity to learn new testing approaches in a safe environment away from daily work, project schedule pressures and software being thrown over the wall.
Continued Page 13

Yes, It's Personal
"I don't want your testers in meetings with my team members. They will be disruptive and then use information they hear to make us look bad. Keep them away from us."
Continued Page 10

Context Driven Jumping
Once upon a time, a herd of sheep were grazing in green fields in the countryside. There was a fence along the field's border to restrict the sheep from crossing over to the other side.
Continued Page 20

Testing & Creativity
The creativity is needed both when generating test ideas outside explicit
requirements, and when finding the effective methods for running important tests.
Continued Page 17

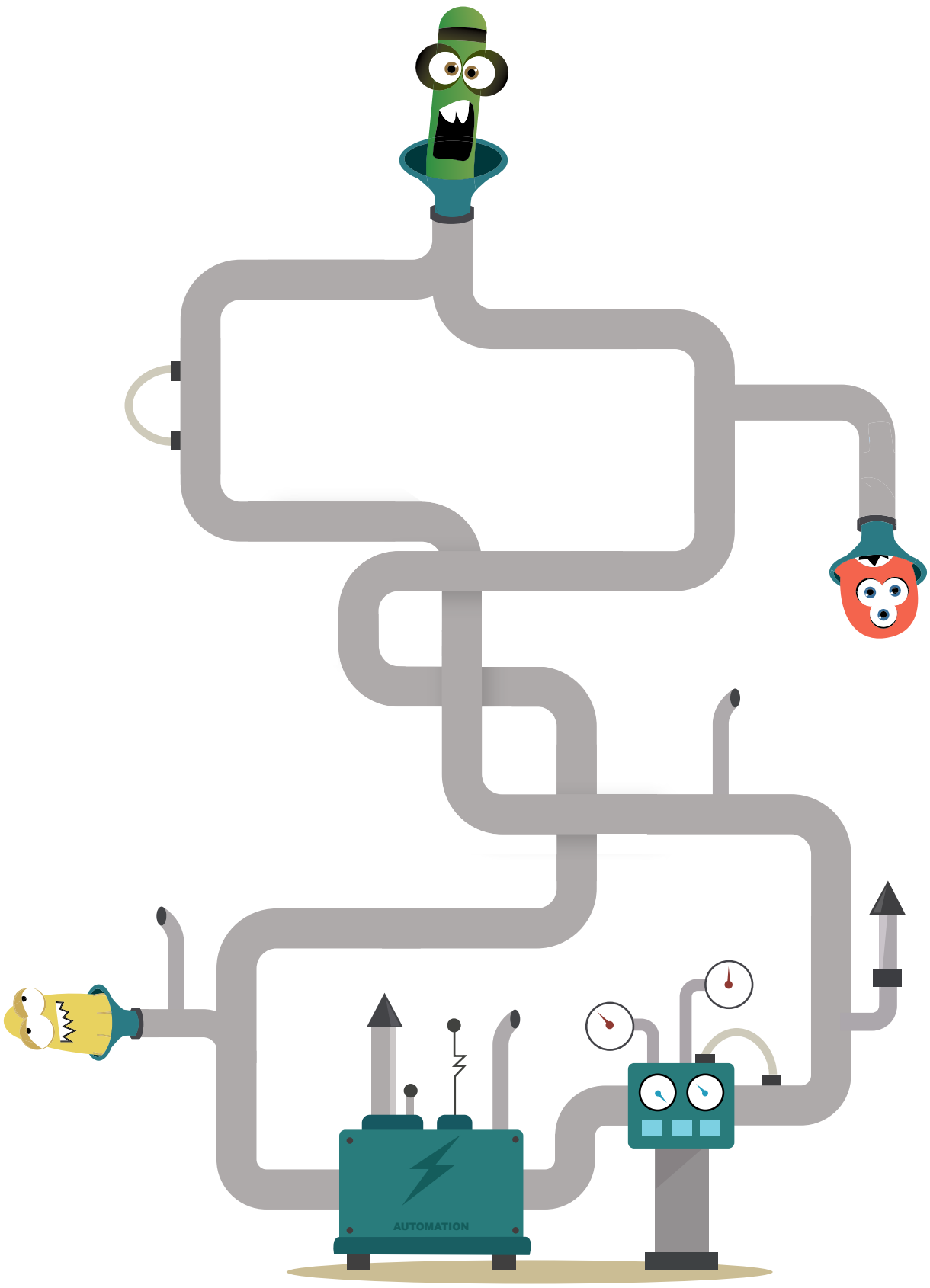
end up with half a million lines of captured scripts (yes, I know of more than one example) that are impossible to maintain.

Even if teams use a framework that allows non-programming members to write tests in plain English, a spreadsheet or some other non-programming language, the test cases still need proper design. I personally have made a terrible mess automating dozens of FitNesse tests where I repeated the same steps over and over, because I didn't know at the time about the !include feature of FitNesse.

Possible Solutions

A simple solution is for experienced programmers to pair with testers to automate tests. At minimum, teams need to hire testers with solid code design skills to help the non-programmer testers. Testers need training in basic design principles and patterns.

There are lots of resources to learn how to do a particular type of automation, such as GUI test automation. My personal mission is to find ways to help educate testers in designing maintainable tests. As a start, I'd like to illustrate some basic design principles using examples written in Robot Framework with Selenium driving GUI tests. I had not used either Robot Framework or Selenium before when I sat down to create these examples. And while I've been automating tests for a long time, I'm not too good with object-oriented programming. If you're experienced with these tools, you may find lots of faults with my test code. But I wanted to demonstrate that these design principles work even when you're unfamiliar with your tool or scripting language.



If you want to play with similar examples, you can download the following:

Information and downloads to install Robot Framework:
http://code.google.com/p/Robot Framework/wiki/Installation

Robot Framework Selenium Library downloads, including the demo: http://code.google.com/p/Robot Framework-seleniumlibrary/

Look up Selenium commands here:
http://code.google.com/p/Robot Framework-seleniumlibrary/wiki/LibraryDocumentation
Run tests with" ./rundemo.py <test file name>"

To start the server for the app under test manually, type "python httpserver.py start"

Example One

We're testing account validation on a web app login page. Our first test will open a browser, go to the main page, and check the title. Then it will type in a correct username and password, click login, verify that the title is "Welcome Page", and that the page contains the text "Login succeeded". (Note: I'd start even more simply in real life – just open the browser for starters. But I have to try to keep this article a manageable length.)

The following is from my test, which is in a .txt file in real life (Robot Framework tests can be plain text or HTML). The asterisks on the section names must start in column 1. There are two spaces between the Selenium keywords such as 'input text', and the data or arguments used such as the field name 'username_field' and the value 'demo'. The "Test Cases" section includes all our test cases. The "Settings" section allows the test to access the correct libraries, and does tidying up.

```
*** Test Cases ***

Test Login
  open browser  http://127.0.0.1:7272  firefox
  title should be  Login Page

  input text  username_field  demo
  input text  password_field  mode
  Click Button  login_button

  Title should be  Welcome Page
  Page should contain  Login succeeded

*** Settings ***
Library  SeleniumLibrary
Test Teardown     close browser
```

Example Two

Cool, we tested that a valid username and password can log in. But we have lots of test cases to cover. Valid and invalid combinations of usernames and passwords, empty ones, special characters, too-long and too-short ones, and more. Repeating the test above over and over with hard-coded values would get out of control pretty quickly. Plus, we want to test in multiple browsers and possibly test multiple servers.

When we see duplication, we always want to extract it out. Good test tools give you a way to do this. Robot Framework provides the concept of "keywords". For example, we can use a keyword that navigates to the login page, a keyword to type in the username, and so on.

Here's a first step towards extracting out duplication in our simple test. We have two test cases: "invalid account", which tests an invalid username and password and verifies the error message, and "valid account", which logs in with a valid username/password combination and verifies the landing page. We've put the values for browser and server into variables. We've defined keywords for actions such as typing in the username and password, clicking the submit button, and verifying the messages.

```
*** Test Cases ***

invalid account
  navigate to the login page
  type in username  invalid
  type in password  xxx
  click submit
  verify the invalid account message

valid account
  navigate to the login page
  type in username  demo
  type in password  mode
  click submit
  verify the valid account message
  click logout

*** Keywords ***

type in username  [Arguments]  ${username}
  input text  username_field  ${username}

type in password  [Arguments]  ${password}
  input text  password_field  ${password}

navigate to the login page
  log  hello, world!
  open browser  ${LOGIN PAGE}  ${BROWSER}
  title should be  Login Page

click submit
  Click Button  login_button

click logout
  Click Link  logout

verify the invalid account message
  Title should be  Error Page
  Page should contain  Invalid user name and/or password

verify the valid account message
  Title should be  Welcome Page
  Page should contain  Login succeeded

*** Settings ***
Library  SeleniumLibrary
Test Teardown     close browser

*** Variable ***
${LOGIN PAGE}   http://localhost:7272
${BROWSER}      firefox
```

This test is still sort of end-to-end-y, in that it first does the invalid case, then the valid one, then logs out. It's a step in the right direction. If the name of the password field in the HTML changes, we only have to change it in one place. Same thing if the text of the invalid login error message changes.

Example Three

It'd be nicer if we could just run the test with whatever username and password that we want to try, and check for the appropriate error message. One way to do this is by passing in variable values with command line arguments. We've refactored the test to expect variables to be passed in. We also separated out verifying the title and verifying the message,

```
*** Test Cases ***

login account
  navigate to the login page
  type in username  ${username}
  type in password  ${password}
  click submit
  verify the title
  verify the message

*** Keywords ***

navigate to the login page
  open browser  ${LOGIN PAGE}  ${BROWSER}
  title should be  Login Page
  ${title} =  Get Title
  should start with  ${title}  Login

type in username  [Arguments]  ${username}
  input text  username_field  ${username}

type in password  [Arguments]  ${password}
  input text  password_field  ${password}

click submit
  Click Button  login_button

click logout
  Click Link  logout

verify the title
  Title should be  ${title}

verify the message
  Page should contain  ${message}
```

```
*** Settings ***

Library  SeleniumLibrary
Test Teardown     close browser


*** Variable ***
${LOGIN PAGE}   http://localhost:7272
${BROWSER}      firefox
```

We can supply the variable values from the command line:

```
./runDemo.py --variable username:demo --variable password:mode
--variable message:'Login Succeeded' --variable title:'Welcome Page'
demo_test2.txt
```

I can easily visualize cranking a lot of username, password and message value combinations through this script. I could write a script to do this so I don't have to type them myself. My team uses a similar technique to test many combinations of data values with our Watir scripts. Different tool, same concept.

It turns out that in Robot Framework, running the test multiple times by passing in variables from the command line would be inefficient, because Selenium would start the browser up new each time. But this can be a good approach with other tools, and works well in Robot Framework for running the same test with different browsers (see http://robotframework-seleniumlibrary.googlecode.com/hg/demo/login_tests/invalid_login.txt for an example of this). Also, passing variables from the command line can be a powerful way to leverage automated tests to set up scenarios for manual exploratory testing.

Notice that I also split out the 'verify title' and 'verify message' test cases. That's a personal preference; I find that if my tests are more granular, it's easier to debug problems with the test itself. I didn't pass in the login page or browser values, but I could.

Example Four

There's still a lot going on in one test file. It would be nice if each of our keywords could be in its own little file. Good test tools allow this, and Robot Framework's implementation is called a 'resource'.

For example, we could take "Navigate to login page" and put it in a separate text file called "navigate_to_login.txt"

```
*** Keywords ***

navigate to the login page
  open browser  ${LOGIN PAGE}  ${BROWSER}
  title should be  Login Page
  ${title} =  Get Title
  should start with  ${title}  Login
```

Now we can simply refer to this file whenever we want to navigate to the login page in a test. In Robot Framework, we do this in the "Settings" section. Notice I also added some Documentation there.

```
*** Settings ***

Documentation  Test account validation

Resource  navigate_to_login.txt

*** Test Cases ***


login account
  navigate to the login page
  type in username  ${username}
  type in password  ${password}
  click submit
  verify the title
  verify the message
```


Timebox the search for bugs or replicating bugs. Use heuristics to help. We can prove the existence of bugs but not their absence. #dttip by @ThomasPonnet

The Keywords section looks the same as before, except with the 'navigate to the login page' keyword removed.

If we're testing a web application, we're bound to have lots of tests that need to navigate to the login page. Now this functionality is encapsulated into one little file. Whenever something about navigating to the login page is changed, we only need to update one file, and all the tests that use it will still be able to do the navigation. We'd want to do something similar with the login account, so that our many GUI tests can simply include the login file in their Resource sections.

Design For Maintainability

I've only scratched the surface of test automation design principles. I didn't even get into patterns, such as "Build-Operate-Check", where we build our test data, operate on it, verify the results, and then clean up the data so the test can be rerun. But I hope I'm getting across my point: successful test automation requires good design skills, and they can be learned without too much pain.

I'm no expert at object-oriented design. My programming experience is in structured and interpreted languages, and with proprietary C-like test automation tool languages. The only OO scripting I've done is with Ruby, but I was able to succeed because my fellow tester was a Perl programmer and all my programmer teammates are willing to help too.

I've shown you some simplistic examples of how I personally would start extracting duplication out of a test script and start forming a library of basic test modules, like navigating to the login page and login. We could have modules for verifying HTML page titles, text, messages and elements, using variables for the names and values.

These concepts aren't easy to learn, but the investment pays off. My team and I have spent lots of time over the years refactoring the early tests I wrote in FitNesse and Canoo WebTest, because I made design mistakes or was ignorant of the tools' helpful features (and I had been automating tests reasonably successfully for at least a decade prior!) As a result of constantly learning, refactoring and improving, we're able to automate 100% of our regression testing without a heavy maintenance burden, and our tests have a high return on investment. When they fail, it's usually because there's a regression bug.

If you've read this far, you must be eager to learn more about test automation design. Ask a programmer on your team to pair with you to automate some tests, so you can learn some new techniques.

Work through a good book such as Everyday Scripting with Ruby by Brian Marick.

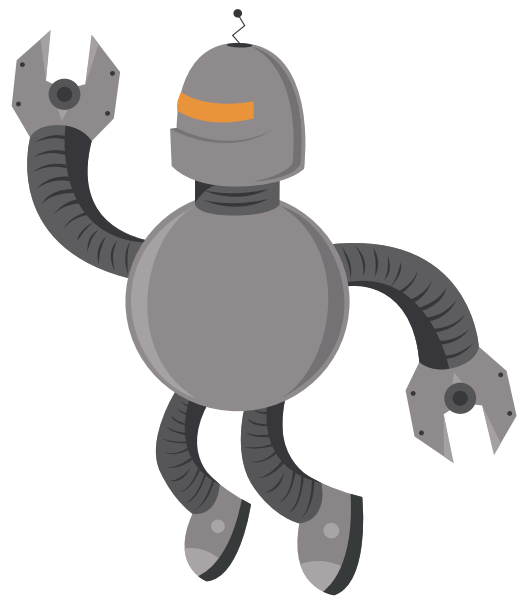
Search Out Blog Posts And Articles.

Dale Emery has an excellent paper on Writing Maintainable Automated Acceptance Tests at http://dhemery.com/pdf/writing_maintainable_automated_acceptance_tests.pdf.

Your team can successfully automate regression tests. You'll need to invest some time and effort to experiment and learn the best approach. It's a journey, but an enjoyable one. Once you get on the right road, you'll have time for the other essential testing tasks you need to create a product your customers love.

About Author

Co-author with Janet Gregory, _Agile Testing: A Practical Guide for Testers and Agile Teams_ (Addison-Wesley 2009)
Contributor to _Beautiful Testing_ (O'Reilly 2009)
http://lisacrispin.com
@lisacrispin on Twitter



Getting a daily dose of Testing Tips

BY ANNE-MARIE CHARRETT

Tuesday has become one of my favourite days of the week because it's when testers from around the globe start posting their valuable and insightful software testing tips on twitter. I'm truly astounded by the depth and quality of these tips.

Daily Testing Tip came about last year as an extension to the Quick Testing Tips website which I had been contributing weekly posts to. I thought that this idea would work well with twitter too, and so Daily Testing Tips was born. I now have over 500 followers and that number grows daily.

I soon discovered that it's quite a challenge to write a daily tip in less than 140 characters and so I put a call out from volunteers. Those who heard the early call were Ajay Balamurugadas, Matt Heusser, Rob Lambert and Trish Khoo.

Many other testers offered to contribute too, so to make it fair, 4 testers now contribute once a week for six weeks.

At the moment I have the fabulous Selena Delesie, Ajay Balamurugadas, Catalin Anastasoaie and Joel Montvelisky contributing a tip once a week. A big thanks to these testers as I know it can be hard work to think up a tip, even if its once a week.

Blog post: Anatomy of a Good Test : http://bit.ly/dAvuhX

Testing Tips

"Think of your testing notes as a flight data recorder. We crack them open if something goes wrong." @jbtestpilot #testing #qa #dttip

Whole team, not only testers, needs to take resp. for quality, testing. "it takes a village". #dttip by @lisacrispin (@lisacrispin)

Learn to be diplomatic, you want the bug to be resolved, not to prove you are right. #dttip by @gunjans

It's not about the test, it's about the result - seek and provide actionable, referenceable information. #dttip by @bradjohnsonsv (Brad Johnson)

Hold your test and automation code to the same standards as you hold the product code. It is your product. #dttip by @cowboytesting (Curtis Stuehrenberg)

Instead of talking about costs in testing, start think about investments, don't translate value in cost, was it worth investing #dttip by @JeroenRo (Jeroen Rosink)

Testing ideas may come to you at the weirdiest times, keep a notebook handy to write them down & use them later. #dttip by @joelmonte (Joel Montvelisky)

Sometimes you have to poke the code with a sharp stick to find out where it bleeds. #dttip by @artofsqa (Troy Hoffman)

One thing that science has taught us some bugs are hard to find but perseverance pays off in the end. don't give up. #dttip by @mpkhosla (Mohinder Khosla)

Automated tests should never be used as a show off to managers. They might think it's easy and useful to automate everything. #dttip by @knorrium (Felipe Knorr Kuhn)

'Ever tried, Ever failed, No matter. Try again, Fail again, Fail Better' Samuel Beckett - thought testers might like it #dttip by @esconfs (EuroSTAR Conferences)

Use typewith.me to collaborate with other testers while pair testing or even with programmers, #dttip by @sdhanasekar (Dhanasekar S)

#dttip the sheer repetition of automated test steps makes subtle errors more visible e.g. when the system counts complex actions incorrectly by @chris_mcmahon (Chris McMahon)

Just because you've counted all the trees doesn't mean you've seen the forest #dttip by @TestSideStory (Zeger Van Hese)

Find all users in a system, not only the ones that people want to know about. That includes disgruntled employee and hacker. #dttip by @Thomas Ponnet (Thomas Ponnet)

As lastactivity for today, start with a short preparation for the next day #dttip by @JeroenRo (Jeroen Rosink)

Assume nothing. Question everything. Trust no one #dttip by @cartoontester

if you want to study the "effect" of something, try removing it and see how the system behaves #dttip by @shrinik (Shrini Kulkarni)

If you test on Windows Vista or 7 be sure to turn on the 'All CPU Meter' windows gadget. #dttip by @can_test (Paul Carvalho)

draw pictures of the software you are planning to test; the pictures will help you think about it differently than the designer did #dttip by @AlanMyrvold (Alan Myrvold)

It's a great experience to post a daily testing tip to the community and all testers are welcome to do so.

So if there are any testers out there who want to try their hand out at writing a daily testing tip for six weeks, send me an email at daily-testingtip@gmail.com with a short bio indicating why you want to contribute. It's that easy!

Tag Tuesday and #dttip on Twitter was an idea from the wonderful Rosie Sherry.

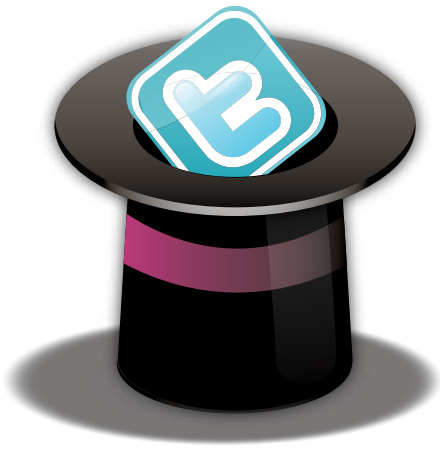
She thought it might be fun to have a day where anyone could share a tip on Twitter using the hashtag of #dttip.

The record so far was 117 tips in one day which was truly amazing and summed up by Markus Gärtner in a tip the next day:

"I think tester's blew the weekly testing tip in a collaborative load testing session on this #dttip hashtag. All after just 3 weeks :)"

Tester had fun voting for their favourite tip in a poll I had on the daily testing tip website but when the tips started getting too many I had to stop that.

I think the success of #dttip is a demonstration of the strength and willingness of the software testing community to help each other out.



I'm proud to be part of that. Also, the beauty of #dttip is that your tip can really help out a tester anywhere in the world. An encouraging tip can help a tester refocus or validate their own thoughts in a very practical way.

I've collated a list of tips that for me stand out (they are in no particular order), I'm sure you have your own favourites.

If you want to post a tip on Tag Tuesday, you will need a twitter account. When you write a tweet, just add #dttip to it anywhere in the message. If you want to follow all the tips, you can either do so at http://dailytestingtip.com or use the feed http://search.twitter.com/search?q=%23dttip

Again, thanks to everyone who contributes to making Tag Tuesdays so much fun.

About Author

Anne-Marie Charrett is a professional software tester and runs her own company,Testing Times [http://www.testingtimes.ie/]. An electronic engineer by trade, software testing chose her when in 1990 she started conformance testing against European standards. She was hooked and has been testing since then. She enjoys working with innovative and creative people, which has led her to specialise working for start-ups and incubators. She's a keen blogger and hosts her own blog, called Maverick Tester. [http://mavericktester.com/]

You know the movie Zombieland? There is a rule: Double Tap! Works for testing perfectly... by @Skarlso

The Managing Tester

PARIMALA SHANKARAIAH

Once upon a time, there was a man called Mr. Smitten who was smitten by testing. His daily struggle to be a better tester kept him alive and kicking with enthusiasm. Whenever he learnt something new and implemented it he would discover that there is still something else to learn. He understood that "Learning is an Ocean" and started savoring it drop by drop even if it would take a lifetime. He seemed to be content with his learning strategy.

One fine Monday morning, Mr. Smitten woke up to pouring rain and set off to go to work. He got ready, beat the morning traffic and reached the office. Unlike other days, there was a flurry of activity in the office. Several managers were running around conference rooms with "long" faces.

The first thought that struck Smitten was "Is another wicket* down?", "Oh No! Not again".

He then got wind that the test manager had resigned for better prospects elsewhere. The test team felt as if it was orphaned forever. Here was a test manager who really cared and fought for the team. He supported and protected the team through thick and thin and now he had quit. It was a big jolt, especially given the series of resignations by managers in recent history.

The senior management of the organization where Mr. Smitten worked was business savvy. They decided not to hire another manager. Their question was "Do we really need a test manager?" Mr. Smitten wondered why this question never came up when the previous manager was doing an outstanding job. The plan was to assign Mr. Clever as the team lead as he was the next most senior person.

They thought that in addition to testing, Mr. Clever would lead the team for no extra cost. So he would work 150% instead of 100%. Sadly, Clever being very clever rejected the offer outright saying he did not like people management. They were disappointed with Mr. Clever and decided not to promote him to the next level and now placed their hope on Mr. Smitten who had great people and communication skills.

They were smart this time around though. They did not ask Smitten about the new role. They dumped that role on him instead. Mr. Smitten did not get a chance to say "No" as he was never asked in the first place.

Mr. Smitten was anointed as the new lead for the test team and took over the remnants left over by the highly talented outgoing manager. Initially he only concerned himself with filling the gaps left over by the previous manager. He had enough challenges already to learn and try new ideas in testing.

However he now had to attend lots of tiresome meetings, interact with other managers and prepare reports as and when upper management told him to do so. Why? Because he was now a manager! Smitten thought he was helpless.

Week 1 - 20 hours spent in meet-

A developer should never test their own software." http://bit.ly/acwzK1 #SoftwareTesting by @teknologika

ings, 10 hrs for reporting and 10 hrs for testing. There was a major release, so he ended up stretching on weekdays and over-stretched on weekends.

Week 2 – 15 hrs meetings, 5 hrs reporting, 40 hrs testing (stretching).

Week 3 – 15 hrs meetings, 5 hrs reporting, 20 hrs testing (No stretching). After all, no one can stretch forever.

Mr. Smitten saw his managing job eating up his testing time which could severely impact the upcoming release. When he raised a red flag, the powers that be asked him why he was not delegating his testing work. Smitten loved to test, not to manage or delegate. He was asked to manage fulltime which he did not enjoy. He understood that management was a new arena for him to learn and experience, but that is not where Mr. Smitten's passion was. His heart belonged to testing. He did not love management as much as testing.

Over a period of time, Mr. Smitten became unhappy. Though he learned to balance management and testing, it was hard for him to devote more time to his learning related tasks. He was expected to know everything about anything that every team member did. He had cursory knowledge of most things, but no deeper technical details. He did not have time to get to the technical level.

He stopped reading technical stuff. He stopped reading testing websites and blogs. He stopped writing about testing. He stopped exploring. He stopped fighting back. He became disillusioned. He was no longer the same man smitten by testing. He started getting a feeling that he was neither a good tester nor a good team lead. His motivation started to sink.

What did Mr. Smitten do? He spoke to his manager the very next day saying he couldn't work this way anymore. He was not interested in a management role. He was interested in testing which he was now unable to do. He felt that he was becoming a failure figure in the team. He poured his heart out. He spoke the truth.

His manager took a while to think about what has happened. Did they do Mr. Smitten a favor promoting him? Or make him a scapegoat like he thought they had?

Would he be forced to remain in the same role, go back to his previous tester role or laid off for not towing the company line?

Did Mr. Smitten do the right thing? Of course he did. No matter the outcome he knew inside that he had stood up for his passion. He had spoken out. He had spoken the truth.

Passionate testers may reject managerial roles not because they lack people skills, but because they don't want to be a manager.

Vipul Kocher (Co President , PureTesting) once said "If you scare a crawling baby by yelling 'walk', 'walk', 'walk', the baby will stop crawling - forget about walking". This sounds so true for testers.

If there are many passionate testers in the team who opt to test versus managing teams, it's not right to force them into managerial roles as a need arises. At the same time, if people are interested in management, but forced to test, they may not do a good job either. Though such decisions have to come at a personal level from each individual, their freedom is constricted due to organizational hierarchies and infrastructure. This in turn will have profound impact on people's motivation.

Passionate testers may reject managerial roles not because they lack people skills, but because they don't want to be a manager. Instead of penalizing them for lack of managerial skills, it would be good to motivate them to work on their technical and testing skills. In fact, it's a great investment strategy to encourage passionate testers because there are very few really great testers. We need more of them.

How many testers have you met who say they have ~20 plus years of experience in testing alone. One, two, three, ten? Just a handful. Let testers grow in numbers. Let testers grow in expertise. Let's hope for a better testing world. Let's encourage passionate testers.

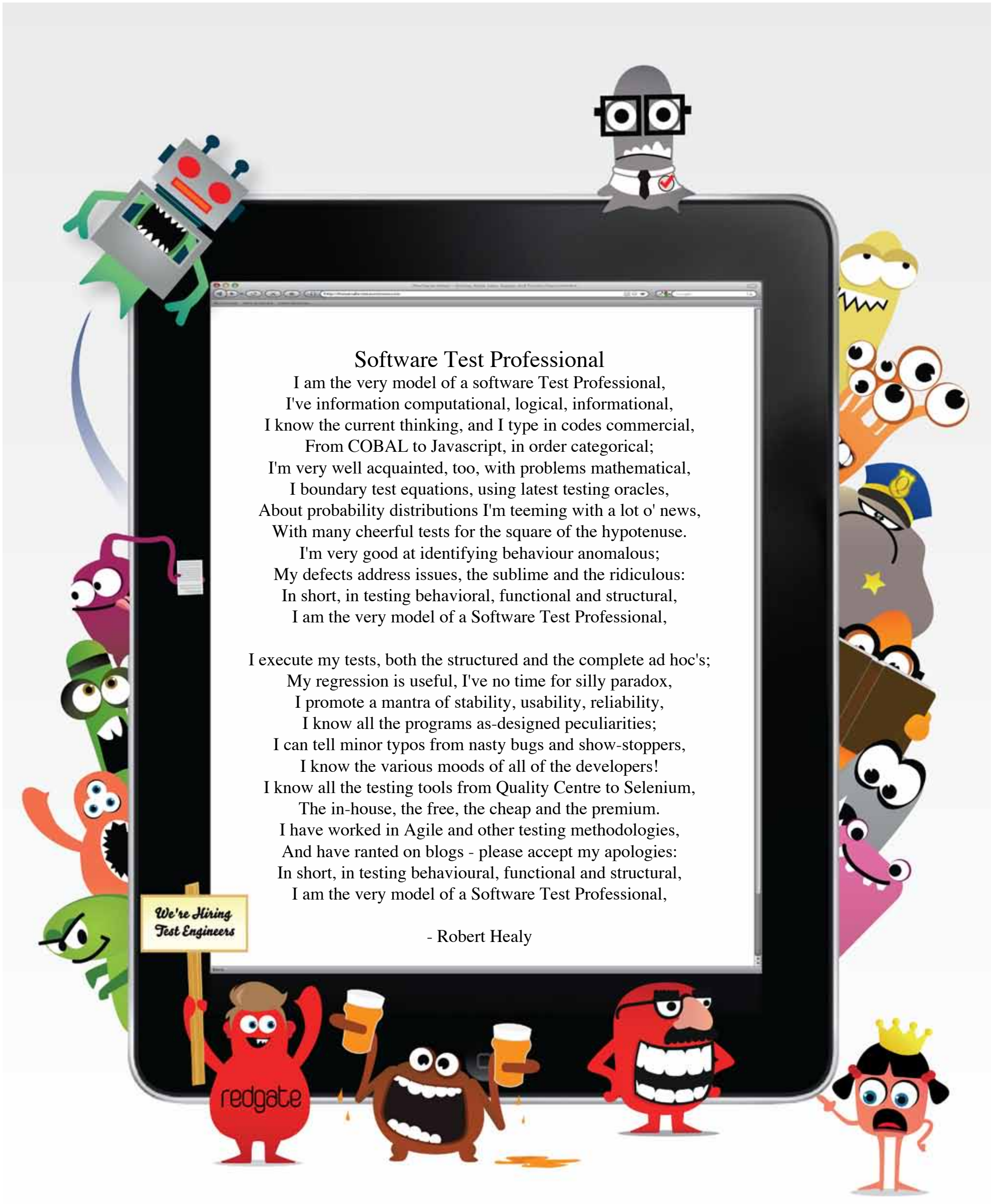
Don't sulk as a managing tester. Rejoice as a testing Tester!

References

Wicket is an Indian word used by cricket fans to refer to someone quitting the organization.

About Author

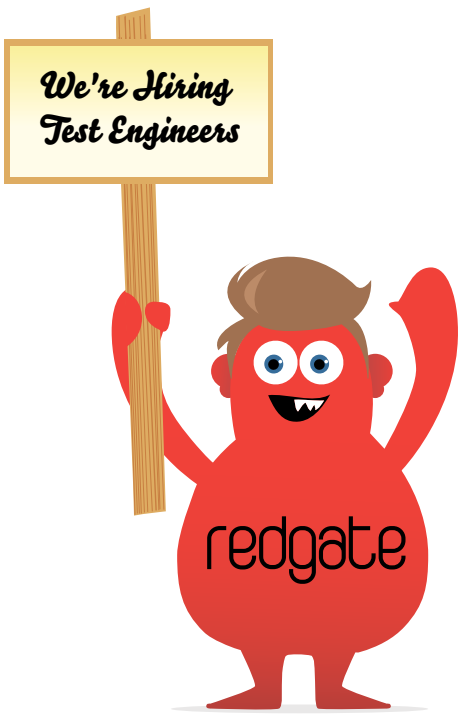
Parimala Shankaraiah is one of the co-founders of Weekend Testing and works at Consona, Bangalore as a senior tester in the weekdays. She handles Public Relations at Weekend testing. She also co-facilitates testing sessions for Bangalore chapter. She authors a blog http://curioustester.blogspot.com/ where she writes about her testing experiences. She can be contacted atparimala.shankaraiah@gmail.com.


Software Test Professional
I am the very model of a software Test Professional,
I've information computational, logical, informational,
I know the current thinking, and I type in codes commercial,
From COBAL to Javascript, in order categorical;
I'm very well acquainted, too, with problems mathematical,
I boundary test equations, using latest testing oracles,
About probability distributions I'm teeming with a lot o' news,
With many cheerful tests for the square of the hypotenuse.
I'm very good at identifying behaviour anomalous;
My defects address issues, the sublime and the ridiculous:
In short, in testing behavioral, functional and structural,
I am the very model of a Software Test Professional,
I execute my tests, both the structured and the complete ad hoc's;
My regression is useful, I've no time for silly paradox,
I promote a mantra of stability, usability, reliability,
I know all the programs as-designed peculiarities;
I can tell minor typos from nasty bugs and show-stoppers,
I know the various moods of all of the developers!
I know all the testing tools from Quality Centre to Selenium,
The in-house, the free, the cheap and the premium.
I have worked in Agile and other testing methodologies,
And have ranted on blogs - please accept my apologies:
In short, in testing behavioural, functional and structural,
I am the very model of a Software Test Professional,
- Robert Healy
We're Hiring
Test Engineers
redgate

Rosie has a look inside Red Gate Software

BY ROSIE SHERRY

There was something that made me smile when I walked into Red Gate's offices last week. Whilst waiting in the reception area I noticed a great Red Gate recruitment poster. It was the cartoon style characters that seemed to catch my eye. The one in the Gorilla suit made me smile.



The relaxed and fun nature of the images reminded me of The Software Testing Club monster style images that we produce. I could sense some strong similarities between Red Gate's and STC's ethos coming along.

It was during our lunch break that I discovered that each of the cartoon characters was actually a Red Gate employee. I wondered if they had a gorilla working for them, but alas, no. Just someone who occasionally dresses up as one!

I was at Red Gate to have a bit of a nose about and get a feel for who the people behind the scenes were. I had kept seeing Red Gate appear in my interweb life. My first recollection was by 'friending' and chatting about software testing with **David Atkinson** - this was over three years ago. I then came across **The Business of Software** - started by the joint CEO. I've been yearning yearning, but unable to go to this one. My strong interest in collaboration and coworking also made me aware of **Red Gate's Startup Incubator.** And then of course, Red

Gate have been advertising testing jobs on STC with free iPads up for grabs for anyone who gets an interview. Unfortunately they've run out of iPads now but they're still looking for 4-5 new Test Engineers. **Here's the advert** on their website and here's a page they've written about **what it's like to be a Tester at Red Gate.**

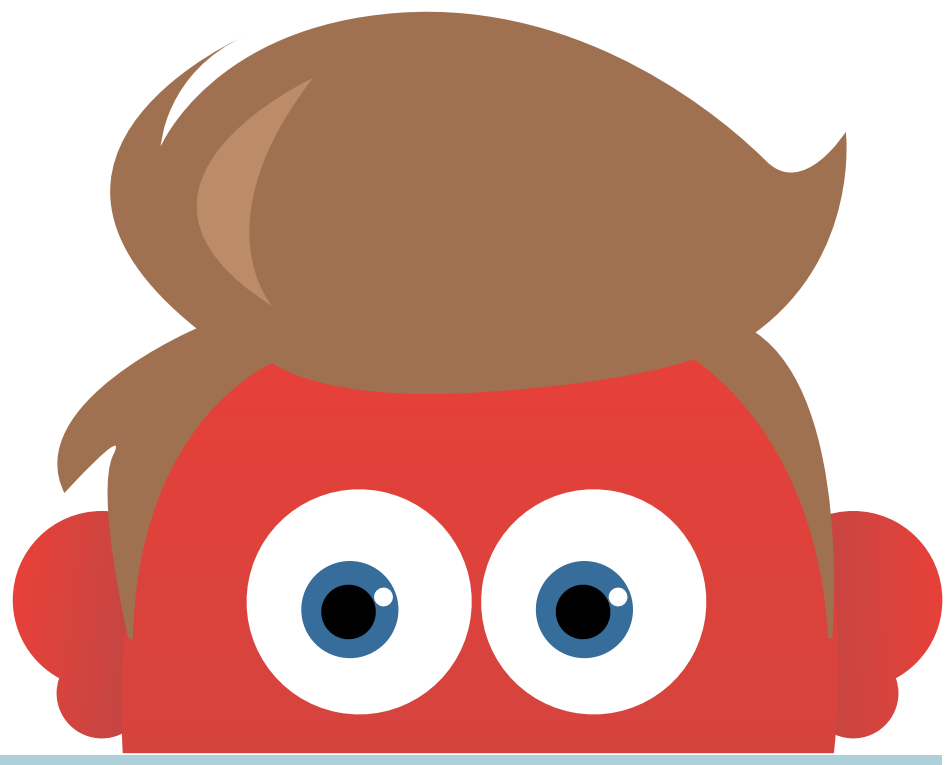
All these reminders of Red Gate's existence had me thinking that it must be a great company to work for. And I admit it. Whilst being shown around Red Gate, casually meeting the joint CEO Neil Davidson, drooling at the employee benefits, having lunch with other Red Gate-rs, viewing how testing is engrained into their agile approach to development and their very open, trusting and collaborative approach to working with people. I even got sucked into their book policy. I love books! This is where anyone can order books up to the value of £100 and claim the money back through RedGate.

Yes, I admit to being tempted to work for them.

I'm sure there are other companies who adopt similar work ethos. Google springs to mind, though they are obviously on a much bigger scale. To me it seems an obvious and logical work ethos to have. To talk about it is one thing, putting it into action deserves credit.

If you are in or near Cambridge and like what we do at STC, then you'll find a similar professional and work ethos to Red Gate. Do check them out.

We're Hiring
Test Engineers



Yes, It's Personal

SELENA DELESIE

"I don't want your testers in meetings with my team members. They will be disruptive and then use information they hear to make us look bad. Keep them away from us."

Fred said this to me a week after we started working together. We had both just joined the company, he as a programming manager, and myself as the test manager. I thought our interactions had been positive, so wondered why Fred felt this way.

Not wanting to make assumptions, I inquired, "Fred, I'm curious to hear about some of your experiences. Have you ever been in a situation like the one you just referred to?"

Looking surprised, Fred replied, "Um, yes. Why do you ask?"

... Does this situation sound familiar to you? Have you had difficulty gaining support from programmers to actively participate in the same meetings and regularly collaborate together?

Building healthy relationships is critical in producing great products. If relationships between different job functions and team members are less than stellar, don't despair. There are techniques to transform hostile organizations into healthy ones, with no management support required.

Listen

It is easy to respond with anger when in a situation where you are intentionally kept out of the loop or unfairly blamed. Anger tends to make situations worse though, and causes problems that need to be dealt with later on.

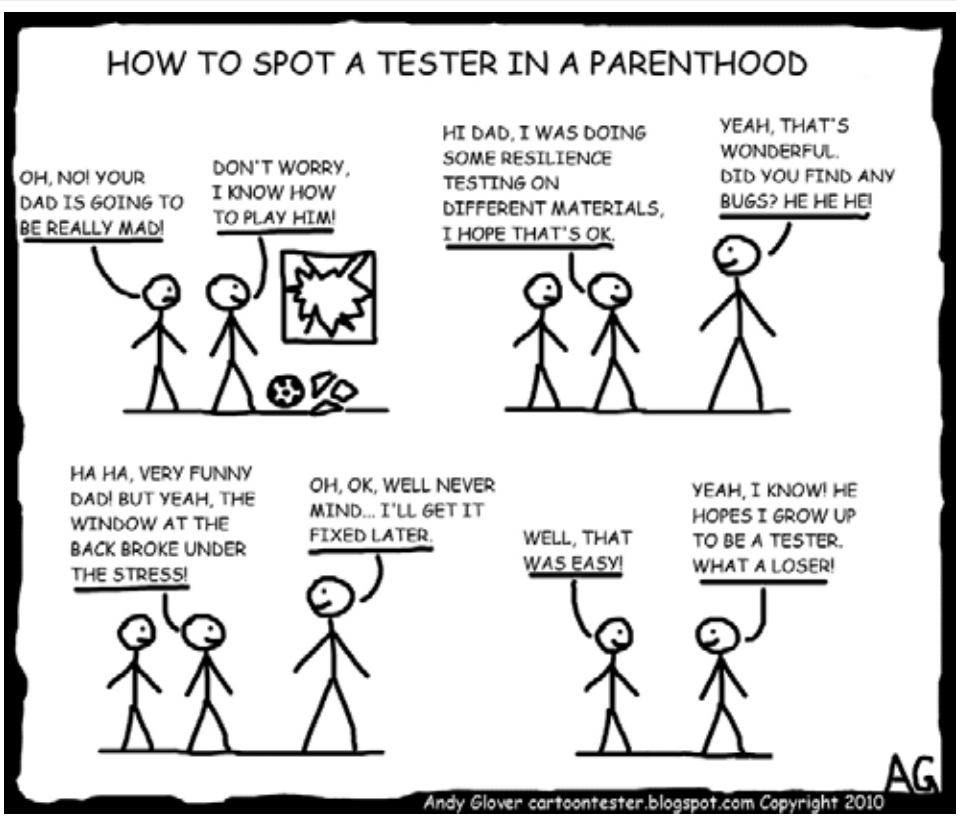
A better approach? Take a deep breath, calm down, and listen.

Understanding the other person's perspective is an important first step in arriving at a mutually agreeable resolution. Allow them to share their frustration then ask them some questions to learn more. Probe gently so you don't come across aggressively.

For Example

• Learn more about their experiences by using 'what, when, where, how,' questions. This helps people open up to a conversation.
• What expectations do they have of the test group?
• What value do they see in the test group?
• How could the test group change to provide more value?
• What do they need from the test group to succeed?

I asked questions like these in my situation, which showed Fred that I was interested in him, his experiences, and his perspectives. This helped Fred be more at ease, as I was not attacking him or defending my position. I was just curious to learn more about him.



Be Personable

It is difficult to maintain a hostile relationship with someone who is friendly and sincerely interested in you as a person. Building positive relationships with colleagues makes work more enjoyable and more successful. This works only if you are genuine in your interactions, as people will sense if you are being deceitful.

It takes time to grow a positive relationship from a hostile one, but stick with it. Working together will get easier, and you may even gain a respected colleague in the process.

Some approaches include:

• Smile in greeting. Say 'Hello', 'Good Morning', and 'Good Night'. People appreciate the gesture, and often can't help but smile in return.
• Use your manners. 'Please', and 'Thank You' can make a world of difference when used genuinely. Let them know when you appreciate work they have done. Your co-worker will be encouraged to work with you, rather than against you.
• Ask about their day or their weekend. Show interest in their life outside of work to start creating a relationship that can be built upon.
• Take them out for coffee or lunch. Spend time together outside of the office to strengthen the foundation for a relationship. At this point you may be seeing each other as real people, not just the annoying person you work with!
• When ready, ask basic questions about their family and experiences. Most people will respond in kind. At this stage you may be building a casual friendship.

In my situation with Fred, I made an effort to stop by and chat with him for a few minutes every day. While he was initially suspicious about my intentions, Fred soon realized that I was genuinely interested in talking with him. He liked being treated respectfully, and not being seen as 'just another programmer'... Now he was a distinct person.

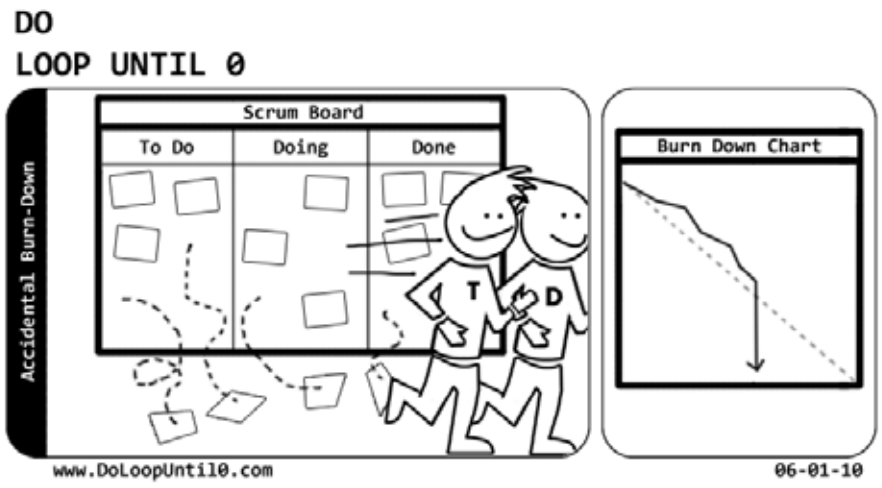
Foster Open Communication

People don't usually want to work with someone they believe to be deceitful. It is easy to be hostile with someone who hordes information, is not open to receiving feedback, or even lies. This does not support the creation of an open environment where people converse and collaborate freely, but encourages a lacklustre environment of negativity.

Creating an environment of open communication is challenging, especially when others behave adversely, but someone needs to take the first steps and lead by example. Why not you? It may take time for your actions to rub off on other people, but they will in time. Some techniques include:

• Speak openly and respectfully about what is on your mind. Share your thoughts with others to help them see that you are not hiding things from them (and you shouldn't be!).
• Focus conversations about problems on processes and projects. While many problems relate to the people involved, a focus on 'fixing' them comes across as an attack. The goal is to engage others, not stop communication.
• Ask for feedback and suggestions from other people then follow through on some that make sense to implement. Others will see that you respect their opinions and trust their judgment.
• When problems arise, ask others if they are open to hear suggestions you have. If they are, share your ideas and suggest a trial run in implementing one. If it does not work out it can easily be re-evaluated or stopped. With this approach you are asking for permission to offer advice and proceed with a solution, rather than forcing one upon them.

My situation required that I first listen to Fred to learn about why he felt the way he did. As Fred became more open during the course of our conversation, I asked if he would



be open to hearing some options. When he agreed I suggested that the testers and programmers attend the same meetings, but that we would discuss and evaluate progress each week. Fred agreed as he was comfortable that we would stop if he was not satisfied. In the end it proved to not be an issue after all, so the joint meetings and collaborations became a regular occurrence.

The Whole Team Approach

Sometimes people forget that each person working on a project is working on the same team. It is easy to niche one another into ''programmers', 'testers', 'project managers', etc. When that happens, blaming other groups for problems can quickly become the norm.

Regardless of job title though, each person working on a project is part of a larger team with the same end goal in mind: Release a quality product within budget that meets or exceeds customer expectations. A shift in mindset is often needed to work together to accomplish the end goal using a 'whole team approach'. This includes:

• Recognize and communicate that you are in this together – no matter what!
• The 'We' mentality – using 'We' instead of 'Me' or 'You'. It is a helpful shift in communication, and people will stop feeling blamed and acting defensively.
• Understand that project and company success is dependent on successful completion and delivery by everyone.
• Communicate and collaborate early and often, which saves everyone time and reduces frustration later on.
• Work through problems as a team, without placing blame. Focus on finding solutions together in order to move forward.
• Do whatever it takes as a team to get it done, regardless of job title or documented responsibilities. Team work is an important part of the job.
•Be appreciative and thankful for the work that your teammates are doing.

In working with Fred, I additionally made a concerted effort to change the language I used when discussing the project and problems. The term 'We' was particularly effective as problems that arose were taken on collectively by the entire team, so no one was put on the spot. Subsequently, whoever contributed to a problem owned up to their mistakes and worked with the rest of the team to resolve it.

Build Trust

The most important factor in transforming a hostile environment into a healthy environment is trust. Trust is the one thing that makes or breaks relationships or change initiatives. It is difficult to acquire and quite fragile. Some key factors to consider in building trust with any colleague:

• Do what you say you will do. It is difficult to trust someone who does not follow through on work they have committed to.
• Build relationships. People are inclined to trust people they know as a colleague, and particularly as a friend. Keeping a distance from others will impede your ability to create a positive work environment.
• Keep private conversations private. Do not become known as a gossip or as someone who cannot be trusted to keep confidential information to themselves. This is one of the easiest ways to lose trust.
• Show competence and intelligence. Unfairly or not, people trust others who show they know what they are talking about in relation to their job. If people believe you don't have the knowledge required to do your job, you will be dismissed and overlooked.
• Follow through in good faith on other techniques described.

In my situation as a new test manager I purposefully continued improving my interactions with Fred and other colleagues. The situation improved dramatically since that first interaction in the weeks following. Co-workers saw that I could be trusted to follow through on commitments, was trustworthy in keeping confidential information to myself, and proved to understand the technology and processes I was working with.

Fred quickly became supportive of my department and I. He and his programmers were soon interacting frequently with us for advice on software design, troubleshooting issues, and processes. Fred also became an advocate for early involvement of the test group in projects, and in needing to hire additional testers to keep up with the heavy workload. I was pleased to have not only gained a positive and healthier work environment, but also a respected colleague and friend.

Conclusion

Regardless of the situation you find yourself in, approach it congruently. There may be something going on for the other person that has nothing to do with you. Learn what you can by asking questions so you don't make assumptions about intentions or reasoning. Do your best to speak openly and honestly with others, regardless of how distressing it may seem.

There are many more techniques for creating an environment where testers and programmers (or any other groups) collaborate and support one another than the ones outlined in this paper. These are a good starting point though, and will make a positive impact.

Remember To:

• Listen
• Be Personable
• Foster Open Communication
• Employ a Whole Team Approach, and
• Build Trust.

Working in a positive and healthy environment with those around you is not impossible. While being an agent of change can be scary, each of us has the power to create such an environment. It can be achieved with diligence, awareness of self and others, and a positive outlook.

Best of luck in creating your own success story!

About Author

Selena Delesie is a consulting software tester and agile coach who runs her own company, Delesie Solutions. She has more than 10 years of experience testing, managing, and coaching in software, testing, and agile practices for leading-edge technologies. She facilitates the evolution of good teams and organizations into great ones using individualized and team-based coaching and interactive training experiences. Selena is co-founder and host for the Waterloo Workshops on Software Testing, and an active speaker, participant, and leader in numerous industry-related conferences and associations. She frequently blogs at www.SelenaDelesie.com.

STC Carnival of Testers June 2010

BY SIMON MORLEY

TESTER'S ACTIVITY FROM THE BLOGOSPHERE

Spring - a time of renewal, rebirth and other re- words... Well, spring 2010 (in the northern hemisphere) has been eventful, diverse and - in the words of Steve 'Interesting' Davis - interesting!

Let's have a look at some of the topics that have been discussed in the last 4 months.

Hot Topics

Spring being a time when things warm up... Let's look at some "warm" topics...

Volcanos

On the lava end of the scale... The general disruption caused by the unpronounceable volcano doing unpredictable things to an unprepared European airspace was a "warm" topic in April. It triggered some systems thinking from Zeger Van Hese: http://bit.ly/awLJRJ, re-arranged schedules for Lisa Crispin: http://bit.ly/9a59Lm and the opportunity for Rob Lambert http://bit.ly/d9RoJs to attend the hastily arranged OpenVolcano event.

Certification

This topic got warm in May. A range of views and discussion was represented on a number of sites. The discussion-and-comment "warm" posts were by James Bach; http://bit.ly/cYojkN Simon Morley; http://bit.ly/cYojkN Nathalie de Vries; http://bit.ly/cPDWTG and Dave Whalen; http://bit.ly/chtOcB

Popular

Some topics have been fun & popular during this period - from tester challenges to weekend testing.

Tester Challenges are a great way to exercise some key thinking and analysis skills for testers. If you can't get weekend testing then having a go at a testers challenge is always worthwhile. If you haven't seen them then it's worth looking at these challenges, there's something to learn in all of them.

Pradeep Soundararajan made a write-up of his "Finding Nemo" challenge; http://bit.ly/9OTpLf

Markus Gärtner used both smurfs; http://bit.ly/abxs9P and a maths proof as subjects for challenges; http://bit.ly/clL1B7

Lynn McKee has produced a range of puzzles from different sources; http://bit.ly/9YOHX1

Lannette Creamer used Star Trek as a base for the challenge and write-up.

A challenge in a bookshop? Parimala Shankaraiah wrote about it, and lessons for testing; http://bit.ly/d6fKOZ

The #parkcalc topic generated a lot of interest in the twitterverse as well as blogosphere. Matt Heusser wrote a review of the challenge and reaction; http://bit.ly/9v73SN

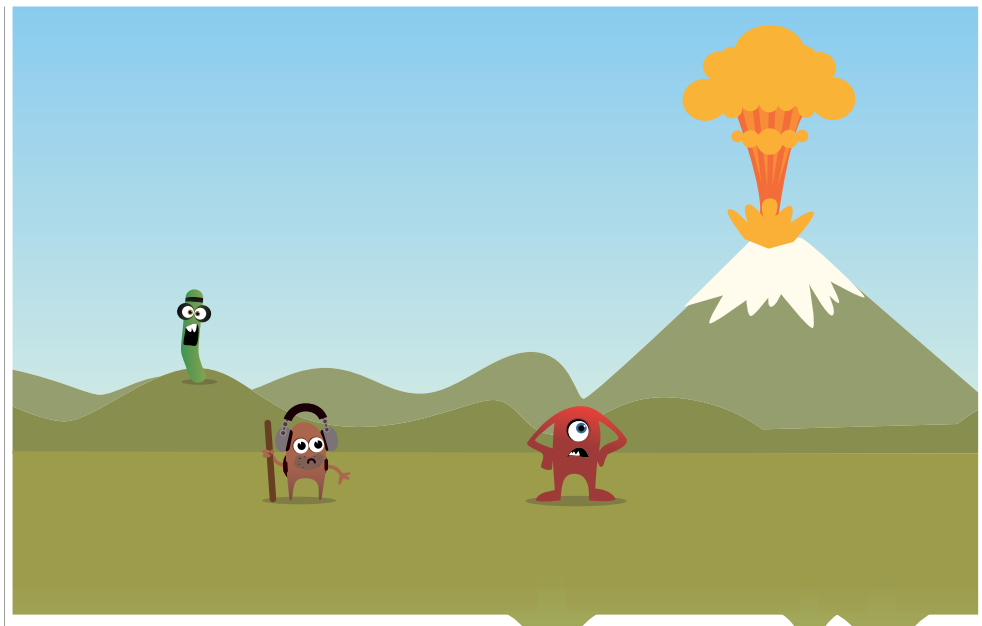
Weekend Testing continued its march round the world by starting an ANZ grouping/chapter. Could it be Africa next? http://bit.ly/aIL4qf

Smorgåsbord

A whole range of other testing topics were up for discussion.

A desire that testers shouldn't be sheep was put forward by Rob Lambert, http://bit.ly/cSDw8e

Ever wondered what Black-Viper Testing is? Then check out Pradeep Soundararajan's post. http://bit.ly/aM3kbN



Different aspects and angles around QA were written about but Michael Bolton's post was the most comprehensive. http://bit.ly/cpLeO2

Toyota's brake-related problems were put into a software testing context by James Bach, here. http://bit.ly/bgYfWP

Reviewing Cem Kaner's description of a good test case was a subject of one of Rikard Edgren's posts. http://bit.ly/bvbmTU

New Pens

New bloggers appeared in the past few months. Here's a selection.

January saw Thomas Ponnet start some intelligent blogging. http://bit.ly/5XE5Zr

Another January new pen was Markus Deibel, with some good reports on his weekend testing involvement. http://bit.ly/5A1cVM

Selenium was the trigger for Felipe Knorr Kuhn to start his blog in May. http://bit.ly/bDIx4d

The analytic approach shines through in James Christie's blog which started in March. http://bit.ly/bdmXJJ

March and April also saw Ann Flismark, http://bit.ly/ap5b19 Tim Riley http://bit.ly/cXLShz & Kenneth Hastings dipping their toes in the blogging water. http://bit.ly/aVC7QM

Leftfield

Analogy is ever-popular amongst testing bloggers. It's helps to clarify (sometimes) and contrast.

Babies and Software Testing was the subject of Elizabeth Fiennes' writing, http://bit.ly/9DSUcv

When was the last time someone called you a muppet (in a nice way?) No? Well, Jared Quinert provides a guide on muppet related testers, http://bit.ly/aQKLs1

Other groupings of testers were given the Animal Kingdom slant by Simon Morley. http://bit.ly/bAuSyr

The near-obligatory Monty Python reference (by Phil Kirkham) is a good way to round off this look back over the last few months. http://bit.ly/9RIfqQ

About Author

Simon Morley posts the Carnival Of Testers on his blog here: http://bit.ly/96ww51

Test Engineers and Development Engineers on the Same Agile Team

BY LIZ ROBERTSON

When my organization rolled out an agile approach across the whole company, we wanted to embrace all the agile best practices, including having each agile team "own" all operations related to completing what they planned for each iteration (including all the testing tasks). For many of the staff, this was the first time that developers and testers were on the same functional team. It became apparent that at least in some of the teams the old distinctions and attitudes between separating between these two roles were still in place.

Given that developers outnumbered testers on the teams, it was often the scenario that stories that were development-complete and waiting for verification would pile up in an iteration, leaving the testers scrambling in the final days of the iteration to verify and get each story to a "done" status. It was suggested by agile coaches that in these cases the entire team should commit to getting stories to done status, and therefore any team member should help get the stories completed, meaning that sometimes developers could test stories (as long as it wasn't one that they had developed).

This is where previously held attitudes about the distinction between developers and testers kicked-in. Was it "beneath" a developer to do testing tasks? Was it a good use of his/her time?

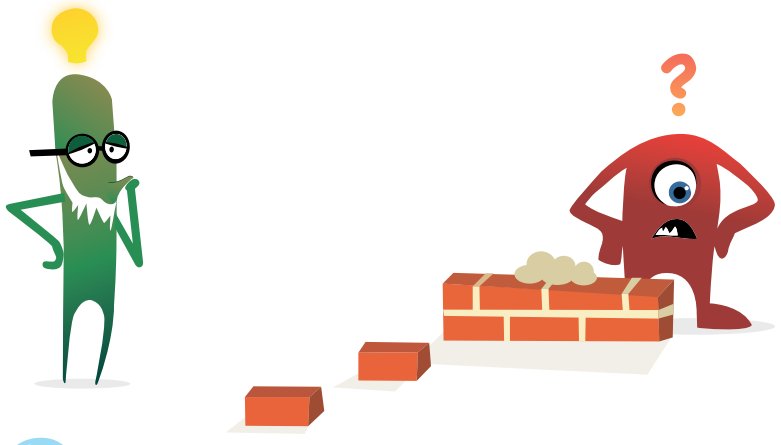
The answers came from adopting a more holistic approach to the overall software development process that starts to blur the lines between what a developer and a tester contribute to a team.

Practices such as test-driven development and unit testing put more testing responsibility into the hands of developers. Expectations from testers on an agile team include working closely with individuals representing the business and with developers to formulate test cases, either manual or automated, which often requires a more in-depth technical knowledge compared to what may have been expected from these same testers in the past.

Clearly the role of the software tester is changing rapidly. As teams adopt more agile processes, as automation plays a greater role in testing, and as flexibility and adaptability are ever more important the expectation of the software tester has shifted from a functional expert who is good at "breaking stuff" to a seasoned technical professional who can provide a myriad of services on a cross functional team to ensure a more consistent and high quality output.

About Author

Liz Robertson is a Quality Engineering Manager for Orbitz Worldwide based in Chicago, IL. Her current focus is on test automation and performance testing, and has been involved with agile development and testing teams for the last 5 years.



Over the Weekend Testing became fun again

BY ANNA BAIK AND MARKUS GÄRTNER

Weekend Testing provides the opportunity to learn new testing approaches in a safe environment away from daily work, project schedule pressures and software being thrown over the wall. Since participation is on a volunteer basis, people feel highly-motivated and eager to learn something new. This article introduces the concept around Weekend Testing, accompanied with comments from the first Weekend Testing sessions.

What Is Weekend Testing?

People might argue "weekends, isn't that supposed to be quality time away from work?" – I admit that this thought crossed my mind too. I now realize that this *is* quality time, and far away from work as well. Quality learning time. - Zeger van Hese

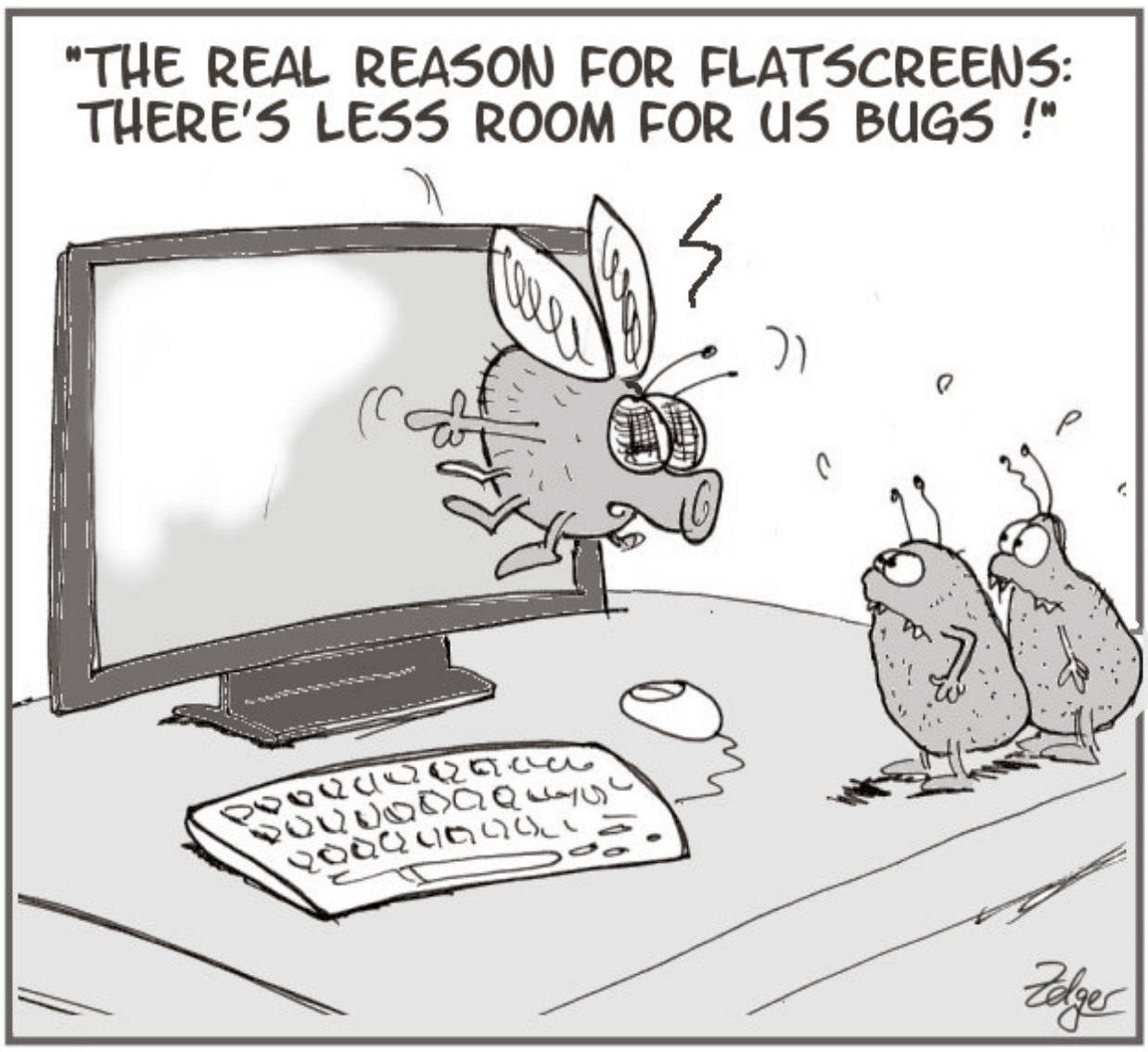
Weekend Testing is a time-boxed, collaborative testing session over the Internet following a provided mission. Sounds very easy. Well, it is easy. The session facilitator leads the session and provides the product and the mission. The participants then start to test the product according to the mission and report their findings back into a public bug tracking system.

Possible missions include finding bugs, evaluating a new test tool, as well as generating test ideas around a given application. Testers often begin by testing the mission itself, by asking questions regarding the mission to narrow it down, or to re-phrase the mission completely - they're wise to do so, as sometimes the mission may contain traps for the unwary and unquestioning.

The session format is a text chat, though participants who've decided to pair up to attack the mission often choose to collaborate, and may use voice chat on a back channel for this. Weekend Testing participants may partition the work among the whole group, or pair with other participants. Being able to test with other people is fun, especially for testers who are the sole tester in their team or company.

The overall session is divided into two major parts. While the first part is spent on testing the product according to the provided mission, in the second half of the session participants share their experiences with their individual testing in a full hour of discussion. Beneath testing traps and bugs, the discussion on individual learning experiences, testing approaches and mistakes as well as tools and techniques are revealed and discussed. Each participant is asked to reflect over the course of their testing. The professional exchange during this part is one of

"Rotate" testers to different projects/products once in a while. It refresh theirs perspective and that of the whole team. @joelmonte



First bug emergency meeting with speakers spreading needless fear

the biggest benefits for each tester participating. Each participant takes away great lessons from the international mix of testers involved.

Testing The Product

... I will say it's the perfect antidote to 6+ months of project death march. Testing is fun again! - Anna Baik

When participants start testing, some may choose to keep a watch on the discussion to see clarifying questions asked to the session facilitators, or to ask them on their own. On occasion, the product developers or representatives may be available in the chat to answer questions about the product, and what information they'd like to know about it. Most of the time, the participants remain silent indicating they're busy testing. Over the course of the first few sessions there have been missions as vague as "test this" up to following a particular set of test heuristics like FCC CUTS VIDS[1].

What To Test

So I am totally sold on the Weekend Testing concept. Great way to try out and learn new approaches [...] and great discussions with fellow testers - Phil Kirkham

Collaboratively testing on the weekend over the Internet leaves little room regarding the products to test. So far, we have tested web applications, which are accessible to everyone around the globe, and open-source applications which may be used on different platforms. FreeMind, Google Calendar, bingMaps, TinyUrl, and WikiOnAStick are just some of the example applications we examined. The session and bug reports from individual testers are publicly accessible, so the product owners can take great value from the tested applications as well.

Watch Out For Traps

...it was simply a lot of fun to be interacting with other testers and finding out how they had approached the tasks and their thoughts on it afterwards. - Phil Kirkham

Think about the last trap you fell into at work - wouldn't it have been better to find that trap when your boss wasn't looking at you? Weekend Testing sessions can make you aware of traps, so that you can avoid them gracefully at work.

Over the course of the first half of the session participants fall into many traps - just as in their day-to-day work. Some fall for the trap of starting testing before the mission was clarified or refined. Some give in to the pressure of the single hour they have for testing - thereby short-cutting either the mission or even the result reporting in the end. Some fall for the trap of the constant distractions caused by the chat window popping up each time a participant raises a question in the group chat. Conversely, some fall for the trap of ignoring the discussion and questions asked by other session members, thus missing out on useful information. Another trap may be that testers decide to work alone instead of pairing up with other testers.

This list may sound familiar. Indeed, these traps are very common for software testers. In our companies we get testing missions like "test this" and while we feel under pressure we jump straight in without stopping to question further how the reporting of the test results will take place or what the main focus of the testing activities will be. Testers who have participated in Weekend Testing are building an awareness of such traps and often take this back to their workplace.

Interestingly, most of the time testers create these traps for themselves and fall diligently into them. So, participating in a Weekend Testing session may help overcome your own self-blindness to these traps.

Discussion

I liked this a lot, as normally I work alone, so I guess this is the closest I get to working in a team environment. - Anne-Marie Charrett

During the discussion and debrief part of the session, each participant takes turns to answer questions on their experiences while testing the product. The facilitator will ask the testers questions to enable self-reflection. While this may sound odd, the debrief of the session is the most valuable part of the overall two hours. During this part participants share their experiences, what problems they run into, reflect over the traps they fell for, and exchange thoughts and solutions collaboratively. Often, each tester may have come up with an entirely different approach to everyone else. This can help other testers to learn about new ways and ideas for their daily work.

In such a deliberate learning environment each participant gets a lot of practical knowledge. While working on the weekends may sound odd, testers are challenged to learn about new domains, new testing approaches and to share and reflect their experiences with other testers around

I learned LOADS from a DB walk through session the other day by the devs. Why not organize something similar with your team? @Rob_Lambert

the globe. Testers get to try new approaches that they don't work with in their daily work. In fact, Weekend Testing provides self-education for testers far beyond classroom-based certification courses.

Global Exchange

I also look at all the participants here and think 'I'm not worthy! These people are all so accomplished!' Then I think 'great, what good company to learn a lot in'. - Anna Baik

Though the sessions are run by local chapters - as of this writing we have Mumbai, Europe, Bangalore, Australia/NZ, Hyderabad, and Chennai - testers globally are invited to participate in every Weekend Testing session around the globe. Every tester participating thereby has the opportunity to learn from basically everyone. Since in the second hour experiences are shared in the discussion, Weekend Testing provides a learning environment beyond traditional classroom courses. A typical Weekend Testing session lasts just two hours, while providing the benefit of globally exchanging thoughts with some really great testers.

Guest Facilitators On Weekend Testing

Over the course of the last year, Weekend Testing has gained the attention of some great testers and test consultants all over the planet. We were happy to bring in guest facilitators into the sessions. Here are some of the experiences with Weekend Testing they made.

Jon Bach

I was invited to be part of two WT sessions (sessions 32 and 33) recently via Skype -- each starting at 2:30 AM Seattle time. With the house dark and quiet, the glow from my laptop was like a cup of ruvenating coffee for me. It was a warm window into what I found to be a collaborative, professional and welcoming enclave of testers. The folks who volunteer their time to hunt for problems and share testing tricks were like meeting the elves and fairies responsible for keeping the world spinning. There's something enriching and soulful watching testers practicing testcraft while testers in US time zones are asleep. The testers I had the pleasure of interacting with in these sessions were indeed magical in terms of their enthusiasm and energy for a profession focused on critique.

Anne-Marie Charrett

If you want a taste at test management, or you want to improve your management skills, I would recommend anyone volunteering to facilitate a weekend testers session. You learn a lot about yourself & testing when you attend a weekend session, but when you facilitate a test session you get to work on those management and soft skills that every test manager needs. Some of the challenges include, creating an idea for a session, sourcing software for the session, planning the scope and the purpose of the session. On the day, you get to practice your communication and leaderships skills. So, if you want to improve your skills further, why not take the next step and volunteer to run a weekend tester session?

Thomas Ponnet

Making time for testing at the weekend is not always easy to do. When my wife asked me "Why are you so keen to test again this Saturday?" I actually had to think about that. I knew deep down that it was the right thing to do. What I said, after some consideration, is that many testers in their day jobs have less than ideal environments when it comes to learning their craft. They might read or hear about something but can't try it as it might endanger their project, or so people believe. The Weekend Testing group is a safe environment where testers can try out new ideas, talk about testing with peers and like minded people at their level of experience. This is very important as often that is not possible in their day jobs.

I feel that I re-discover my enthusiasm for testing whenever I'm talking with peers, be it at conferences, tester groups or [European Weekend Testers] EWT. In my book it's an important part of why I do my job, to learn, confirm assumptions and things we believe to be true. Another important point for EWT is that at the weekend sessions people may make mistakes. The only thing that happens when they do is to learn from it and get encouragement from the group as many people will have done the same ones themselves.

When I started with EWT I participated only. I'm now moderating as well which has its own challenges and learning curves. It's fantastic to observe, from a test managers perspective, what works in a group and what doesn't. EWT is not only great if you want to learn about software testing but also how to teach effectively, how to steer a discussion and much more. Some great people gave up their spare time to help me in the past so it feels good to give something back now. I might not be there every week as my son wants a part of me as well. But I'll make sure to participate and teach and learn whenever I can.

References

[1] FCC CUTS VIDS heuristic http://bit.ly/9ezwWZ

About Author

Markus Gärtner is a software tester since 4 years from Germany Personally committed to Agile methods, he believes in continuous improvement in software testing through skills. Markus presents at Agile and testing conferences, blogs at http://blog.shino.de and is a black-belt in the Miagi-Do school of software testing, and a co-founder of the European chapter of Weekend Testing.

About Author

Before testing, Anna Baik taught telepathy to telephonists, teaching people how to figure out how to ask their customers the right questions, and how to really listen to what they said - and didn't say. She then got a job in application support, and was drafted into testing from there. When she realised testing was about asking the right questions, and seeing what isn't there, she was hooked. She blogs occasionally at the Software Testing Club, and tweets as TesterAB. She and Markus co-founded the Europe chapter of Weekend Testing.

Your brain also requires a reboot every now and then. Don't take long testing runs without a rest or they won't be so productive. @knorrium

Why Testers Should Care About Static Analysis

BY ANDREW YANG AND FLASH SHERIDAN

Static source code analysis ("static analysis" for short) tools have been around for a long time. Nearly everyone has heard of Lint, a program from the 1970's, which flags potential problems in source code, along with a large number of non-problems. Static analysis tools advanced far beyond Lint in the past decade. Modern day static analysis tools perform sophisticated data and path analysis across functions and can detect problems with far more accuracy and depth than traditional tools.

Organizations intent on improving their software have embraced static analysis as an important part of the software development process. Spurred on by the increase in popularity of Agile Development, the increasingly visible costs of software failure, and other trends, software development organizations are automating their testing processes and finding and fixing errors and security violations earlier.

Static analysis finds defects by analyzing source code without running it. No test cases are required and a good static analysis tool like Klocwork or Coverity can analyze large numbers of logical paths in the code to find plenty of serious bugs. Static analysis tools don't even need fully working software to operate, allowing programmers to use it from the early stages of coding. Feedback is quick and errors are reported pointing to the specific line of code where the potential bug is, making it easier for programmers to fix. If a bug is found by static analysis rather than testing, it saves almost all of the programmer's time in isolating and diagnosing the problem, and all of the tester's time in reproducing and reporting it, increasing the productivity of both.

Actors

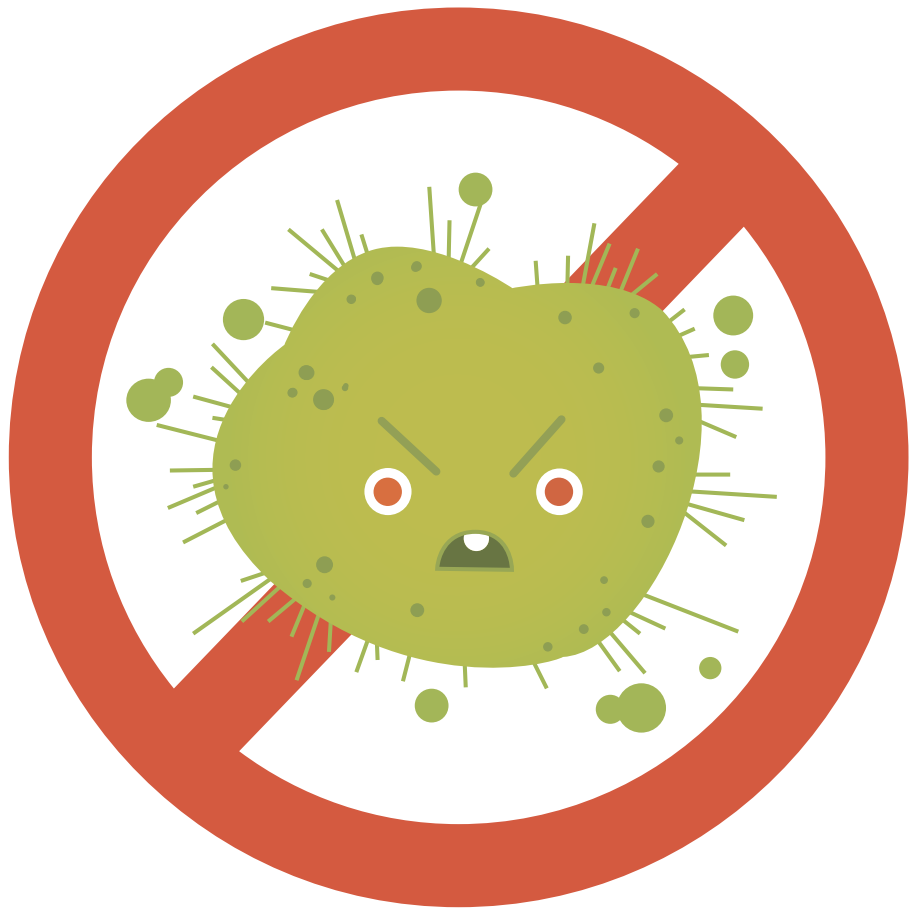
Who gets involved in static analysis? Usually architects and programmers are the drivers for getting a static analysis tool. They realize that finding problems while they are coding is the least expensive way to improve quality and security. In an organization large enough to have a tools team, it may end up owning the implementation and maintenance of the tool. Other groups like central security or governance may play a role. Testing should play a pivotal role but usually does not.

Why Testers Need To Care

Testers should be involved in and help to drive quality and security throughout the entire software development lifecycle. Gone are the days where the tester's sole role is to do the last minute testing just before release.

Testers serve as a strong counterpoint to programmers, creating a healthy dynamic towards improving quality and security. As the part of the organization focused primarily on software quality, testing's role is to ensure that quality standards, and not merely the smooth running of the tool, are actually achieved.

One of the most important roles in any static analysis initiative is finding the person or department that actually cares that the right defects are getting fixed. Quite frequently, without the right processes and advocacy, the tool becomes limited in its use. For instance, developers who are in charge of prioritizing their own defects may incorrectly mark genuine bugs as false positives. Programmers have a lot on their plates and taking the time to review the problems accurately naturally becomes de-prioritized. Checks and balances are needed.



Testers can play an important role: defining and helping to enforce the goals for static analysis. Must all static analysis defects be fixed before release? Should "no new defects" be the goal? What are the priorities for each defect category — should all crashing bugs, overflows, memory leaks, and uninitialized data be fixed but other, lower priority, categories like dead code be handled later?

In an organization large enough to have a software engineering or tools development group, one or the other will generally be in charge of deploying and running static analysis tools. This is natural for the deployment phase for the tool; but once the tool is successfully deployed, the goal changes, and testers need to play a larger role. The software engineering group is primarily interested in developing code, and the tools group in having the tool run smoothly in accordance with a clearly defined procedure. Both attitudes conflict with the goal of finding bugs and getting them fixed. Bugs are unpredictable and messy, and finding them without getting inundated with false positives requires flexibility and judgment rather than adherence to a predefined procedure.

As an example, the problem mentioned above, of programmers incorrectly marking their own defects as false positives, has no simple procedural fix. Simply forbidding programmers to mark defects as false positives, or forbidding (or even delaying) configuration changes to eliminate them, will bring the static analysis tool into disrepute. It can even make the code worse, by forcing an inexperienced programmer to change code in a hurry merely to silence the tool. Any genuine solution will require judgment and prioritization of individual defects, which is a matter for testing to drive to resolution. It may not be necessary for testers to review all alleged false positives, but at a minimum, the definition of what is acceptable and how it will be assured must be made.

Improving software development quality early in the process means that testers are testing better software. Fixing the problems sooner helps testers focus on more functional testing, where they can add significantly more value.

How Can Testers Get Involved?

Test teams now require more skills — they need skills in security processes, automation, and even in programming. Testers should understand static analysis technology and the benefits it can provide. With this knowledge, testers can work with other stakeholders to obtain buy-in that it will benefit a larger quality

The real danger isn't that computers will begin 2 think like men but that men will begin 2 think like computers.
~SydneyHarr... @AGareev

and security goal.

Testers need to view quality as a whole, which includes building quality and security into the software development process. Test teams can help define reasonable acceptance criteria and work with development and management teams to create the right processes and infrastructure to support the effective use of the tool. When properly implemented, static analysis will result in big gains in productivity as well as quality.

About Author

Andrew Yang and Flash Sheridan are consultants with Code Integrity Solutions (http://www.codeintegritysolutions.com), a professional services firm focused on static analysis and software development infrastructure. Code Integrity Solutions helps leading companies greatly improve the quality and security of their code. Andrew and Flash can be reached at andy@codeintegritysolutions.com and flash@codeintegritysolutions.com.

Testing & Creativity

BY RIKARD EDGREN

I think I love software testing because it involves such a great deal of creativity - "the process of having new ideas that provide value". Sir Ken Robinson. [http://www.ted.com/talks/ken_robinson_says_schools_kill_creativity.html]

The creativity is needed both when generating test ideas outside explicit requirements, and when finding the effective methods for running important tests.

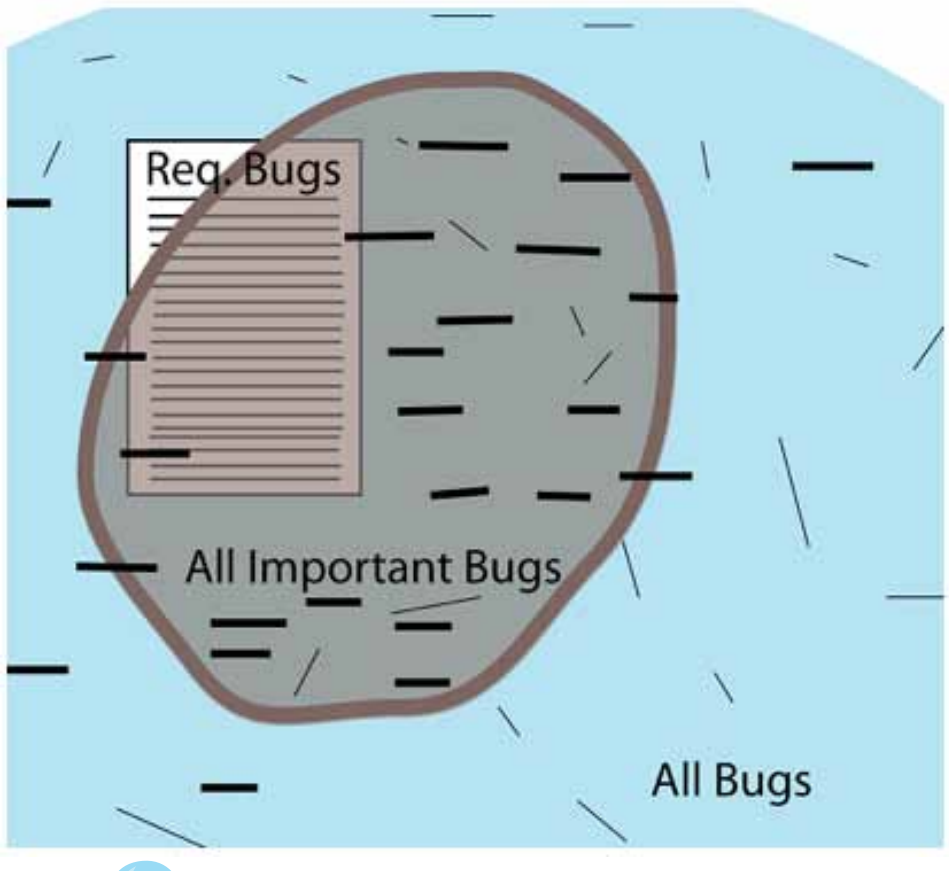
I'd like to explain this by visualizing software in the form of a potato The square is the requirements document, which covers important things that are easily expressible in written format. The implicit requirements, things that a user would want, are not included here.

The blue area is all possible usage and errors, which in reality is an infinite area.

But we're lucky! The area for important things isn't that wide, and we can perform sample tests (symbolized by dashes.)

The samples doesn't cover everything, but if a skilled tester looks at what's happening, there is a great chance that they will discover important issues in nearby areas (this is the everyday serendipity of the software tester.)

Creativity is needed to go from



the requirements to "what's important", and to do it in an effective way.

Testers can combine knowledge from a variety of models, from the product's history, from the technologies in use, from software testing itself, from actual usage (or stories thereof), from the context of the project, by conversations with different people, and by using their own skills.

We can't find every bug, but we can aim to find all bugs that are important.

Cheating With Creativity

Since each project is unique, I can't give you any secret methods of how you can create low-hanging fruit with ingenious test ideas and approaches.

But I can give you some hints on cheating:

Allow Creativity And The Mistakes That Go Along With It

Try many approaches, and cheat by letting there be some waste in your test efforts. Try rotating a "free role" within the team. You will make up for it, eventually.

Have a diversified team, and let people collaborate and inspire each other. Trust the tester's intelligence.

Use Check Lists And Cheat Sheets

Your own list is the best, but as inspiration, search for "Test Heuristics Cheat Sheet", "Exploratory Testing Dynamics" or "Software Quality Characteristics"

Get inspired by solutions and bugs for similar, and radically different, projects.

Count On Serendipity

You will find the best things when looking for something else; execute two different tests at the same time, and explore the areas in between Keep all your senses wide open!

Learn A Lot, And Combine Disparate Knowledge

Creativity never comes out of the blue; it is most often a new combination of existing things.

Ergo; learn more different things, and combine them in fruitful ways.

Steal, combine and adjust two well-known concepts and invent your own method.

Use Creativity Techniques

There are many ways to stimulate creativity, e.g. brainstorming, provocative operators, mind mapping, pair testing…

My favorite is "the opposite method"; thinking about things like "How can our software be as bad as possible" (Can't start it; Can't Install it; Can't Find it) has a good chance of generating insight of what is really important.

Creativity is not a gift, but rather the result of discipline and patience together with an open and curious mind.

Everyone is creative. Find your creative thinking and set it free.

About Author

Rikard Edgren, tester at TIBCO Spotfire, blogger at thetesteye.com

Eat lunch with your developers -- they'll talk about changes they've forgotten to tell you about. @jbaibel

Black Box Software Testing

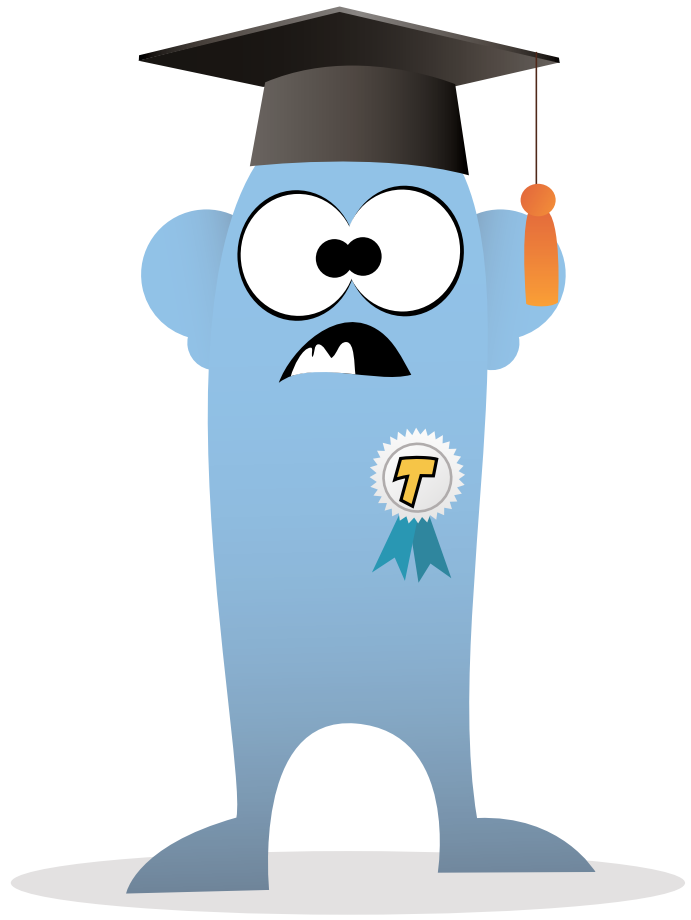
BY SCOTT BARBER

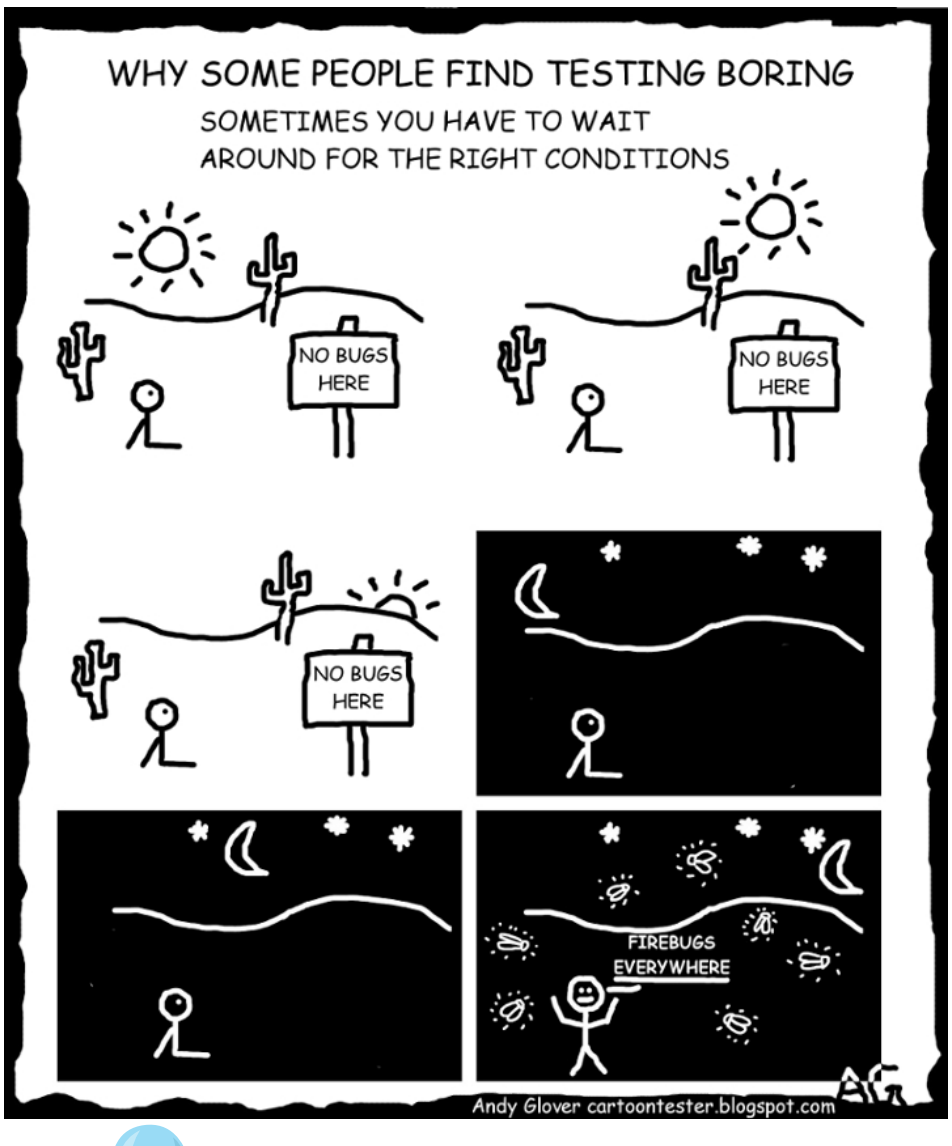
Depending on what research you trust, there are between 4 and 14 million people currently employed in the areas of software quality or software testing worldwide. I like to say "about 10 million". Of them only a tiny percentage, I estimate in the neighborhood of 1%, receive any reference-able training, professional development, or education specific to testing software each year. If total book sales are a fair indicator, fewer than 10% those employed in software quality or software testing own a book directly related to their job. With numbers like this, it's no wonder that many people feel like education and training in the field is, shall we say, lacking.

And what about the quality or effectiveness of this existing education and training? We know that the majority of the available education and training is delivered by tool, certification, and training vendors who, appropriately, are as interested in promoting their products and earning a profit as they are in the quality, breadth of applicability, and effectiveness of their training. When taken together, it is clear to me that testers should have more access to high quality, affordable, broadly applicable, and reference-able education and training.

Over the last 3 years the Association for Software Testing (AST) has been piloting a new way to deliver education and training to testers that we refer to as the BBST program. I'm sure that many of you have at least stumbled across something about this program in the blogsphere, but I'd like to take a few minutes of your time to share some things about it that you probably don't already know.

Understand that my intent is not to "sell" you on this program (though I would be pleased if after reading this at least some of you decided to take one or more of the courses). My intent is to share with you what I believe is the best currently implementable model that I am aware of to better educate more testers at all stages of their career and to raise your expectations about





the education and training that should be available to all testers.

Also understand that it's not my intent to discuss the quality of the content of this, or any other, course. While I believe that quality education and training begins with quality content, I also believe that it takes more than just content for people to learn effectively. What I am discussing is how a particular delivery format is being shown to be more effective, particularly for working testers seeking professional development, that either university or commercial formats you may be familiar with. In fact, the content that is currently available through the BBST program has been, and continues to be, delivered in both university and commercial formats, and that is part of how the effectiveness of this format has been assessed. But don't take my word for it, read what this student had to say in the mandatory, but anonymous, post-course evaluation survey:

"This was a terrific experience. I really enjoyed it. I learned a lot. I got to be around diverse testers from diverse backgrounds who came in from all over the world. I had some really knowledgeable testers give me advice and correct me, which made me learn more than I thought I could."

I've delivered a lot of training in my career, and read a lot of highly complementary course evaluations, but I've never seen the words "… learn more than I thought I could." I struggle to think of higher praise. It's my opinion that this kind of praise is only possible because of this program's unique variety of teaching, learning, and experiencing methods employed during each course. The following description of the program comes from the latest status report to the National Science Foundation:

"Each of the AST courses is a fully-online course taught over four weeks. The first three weeks use a collection of videos, readings, online activities, reflections, and quizzes to present course materials. In the fourth week, students turn their attention to writing a final exam, peer reviewing another student's exam, and completing the course evaluation. In addition to the formal instructional delivery, we provide online space for student study efforts (asking for help, discussing quiz results, and preparing for the final exam) and social space to meet their peers and share non-course related information regarding jobs, promotions, new babies, and links to interesting current events. For more details, please refer to the complete paper at http://bit.ly/cYIGwg.

Assignments are due twice a week, on Wednesday and Saturday. Over a 4 week period successful students spend 32 hrs or more per-

use personas, aligned with the sequence in which they'll do things in the app, to think of scenarios to explore and test. (#notnew) @lisacrispin

forming all of the activities above at a time when it's best for them. Unlike a 5 day face-to-face class that also provides approximately 32 hours of interaction with the material, this format allows students to "pause, rewind, play, and replay" at will to work around distractions, to take some time to absorb concepts, and to discuss the material with their peers, instructors, and co-workers at their own rate. Students don't have to worry about having their questions "tabled" because they don't fit in the course schedule. Students can even "test" the material as they go at their workplace instead of having to wait until after the course is over and the instructor is gone to figure out what implementation challenges they are up against.

Other things that distinguish these courses from other formats with a similar number of hours of interaction is that in addition to video lectures, readings and discussions, it has individual and group exercises, welcomes discussion of diverse experiences and opinions, and employs self, peer, and instructor assessment. In the words of another student:

"In my opinion, the 4-week long online course was a systematic journey towards learning a few core fundamentals of software testing. In addition, it helped me build a practical approach to look at test strategies, testing mission and oracles, and how the mission is dependent on context. This journey was very effective because it was based on cognitive learning -- due to which the knowledge could be applied and used in new situations while thinking about new challenges. I think that this was the epitome of this online training program."

It is the diversity of methods that enables courses in this program to focus on problem solving and decision making while introducing the student to new material rather than simply presenting disembodied procedures to accomplish specific tasks that may or may not turn out to be at all applicable when you get back to the office, or as this student writes:

"This course forces the participants to think. That immediately differentiates it from commercial courses. I'm only slightly kidding. The format also challenges the participants writing skills when answering and reviewing other students' work. This is a skill that I find more and more important in the relevant professional world."

This student was particularly impressed by the value of learning in an environment where they can interact with a diverse peer group:

"One of the best things about this class was the peer reviews and being able to work with people on the assignment. It was amazing to me how relatively simple it actually was. People were civil and kind, and yet could assess your work and be honest. I really loved being around other software testers who really love what they are doing."

This course forces the participants to think. That immediately differentiates it from commercial courses

The following students draw comparisons to other training they have received:

"This course was more intense and made more sense than {certification} (which I have done) - its more in depth and had me thinking at a different level. I have identified some of my weaknesses and will be working on them. The BEST course I have taken on software testing - period!"

"I have taken a number of commercial courses and they gloss over a lot of what I have learnt in the last 4 weeks. BBST is worth more than the {deleted} certificate I have in my possession!"

"I feel you need only one skill to clear commercial exams, that is to memorize and deliver. University exams, though, depended much on memorizing. They had labs, group discussions, and assignments, which was more challenging and helped [me] to learn, but I somewhat felt the application was missing, which was addressed in this course. The examples discussed, peer reviews and the orientation exercises were fantastic in BBST and I feel all these are missing in the university syllabus."

As an added benefit, all of the BBST course materials are published under a Creative Common's License, meaning that anyone is welcome to use the same materials to conduct their own training, with proper attribution, of course. This might not make a big difference to students while they are taking the course, but I know that when I started filling lead and management roles, I found myself wishing that I had access to materials from courses I'd taken years before to help me train my staff. I also know that even when I knew people who were willing to pull strings on my behalf, never once was I successful in getting my hands on anything more than the slide deck.

Did I mention the pricing? AST offers Foundations free members and charges $200 for members ($300 for non-members) for advanced courses. I think you'd agree that compared to most reference-able training available today that this qualifies as affordable.

163 students have successfully completed the first course in the series (Foundations) as of June 1, 2010, and 48 have completed the second (Bug Advocacy) with more classes currently in session. There are a number of new courses in active development, a growing number of trained instructors, and growing interest from potential delivery partners. The next face to face version of the instructor's course (open to anyone who has completed one BBST course) will be held on August 5, 2010 in Grand Rapids, MI, USA for a mere $100 ($50 for AST members) on the day following AST's annual conference, (see http://www.CAST2010.org for more information). Those numbers certainly don't put much of a dent in millions of testers who didn't receive any training last year, but it's a start. And the numbers are growing. For more information visit the links below:

AST's delivery of BBST Courses: http://bit.ly/1a5nMj

BBST Instructor Training: http://bit.ly/awZ0r5

Course Materials: http://bit.ly/1KijAa

Whatever you may think about AST's program, I ask you to put the training model to the test against your own training experiences, then ask yourself the following questions:

• Why it is that most software testers don't receive any training at all each year when more and more fields all the time require people to complete some form of professional development annually?

• Why it is that so much of the training available for software testers is reported to be "not useful", "not implementable", or "not permissible" for their job or their company when asked about the training later?

• Why it is that so much of the referenceable training for software testers is so cost prohibitive?

• Why have we accepted this as the state of our field's professional development?

I suspect that after asking yourself those questions, you'll find yourself asking yourself one more question - the question that led me to get involved in this program in the first place…

• What can I do to improve the availability and quality of training available for people working in the areas of software quality and software testing?

Maybe the answer to that question will lead you to get involved with the program too. Maybe it will lead you to start a training program of your own. Maybe it will lead you to speak out against over-priced, ineffective, low quality training that you've experienced. But if you agree that training and education in our field is lacking, I want to encourage you to take action.

I do not believe that training and education in our field is going to get better on its own. I believe that it is going to take testers and their employers demanding something better and rejecting training that isn't meeting their needs or isn't cost effective before we see widespread improvement. I believe that better and more accessible training is one thing that testers can actually unify over.

As a field, we may never agree on what a test case should include, what percentage of tests should be automated, or how many testers per developer is ideal, but I see no reason why we can't unify over demanding, creating, and making widely available affordable, high-quality, referenceable, effective training and education for software testers – Do you?

About Author

Scott Barber is the Chief Technologist of PerfTestPlus, Executive Director of the Association for Software Testing Co-Founder of the Workshop on Performance and Reliability and co-author of the Microsoft patterns & practices book Performance Testing Guidance for Web Applications. Scott thinks of himself as a tester and trainer of testers who has a passion for testing software system performance. In additional to performance testing, he is particularly well versed in developing customized testing methodologies, embedded systems testing, testing biometric identification and personal security systems, group facilitation and authoring instructional materials.
www.perftestplus.com

Context Driven Jumping

BY DHANASEKAR SUBRAMANIAM

Once upon a time, a herd of sheep were grazing in green fields in the countryside. There was a fence along the field's border to restrict the sheep from crossing over to the other side. One fine day, one sheep from the herd jumped over the fence to the other side. Another sheep followed. The shepherd saw this and decided to remove the fence so that the sheep would not get hurt.

None of the remaining sheep realized that the fence had been removed and continued to jump over a fence that did not exist anymore.

Why not use something that has succeeded in the past?"

This type of herd behavior is not just restricted to sheep, but also among us testers as we try to follow the so called processes, best practices and industry standards. Such standards often start as an answer to a problem which itself has a specific context. Yet many testers who follow these processes don't bother to check if it suits their context or not. If a particular process/best practice/standard has succeeded on a couple of occasions, it is assumed that it will succeed for all time.

Few people analyse these practices or standards to see if it suits their context. Few people revisit these practices or standards and question them. Few people trial these practices or standards before applying them to their own projects. Suppose, there was a team leader who said you should execute 23.45 test cases per day, find 23.456 bugs per thousand lines of code and automate 3.56 test cases per day. The next day this becomes a standard for every tester in the team. It becomes a Best Practice.

I once worked for a company which asked testers to execute X number of test cases per day. Executing X target would ride so high on tester's minds that they would ignore all the bugs that they find outside of the test cases, assuming that any of the testers ventured from the scripts in the first place. Their focus was on X test cases in a day, nothing more, nothing less. Such standards cage testers' imagination and prevent them from thinking beyond the test scripts.

If such standards are questioned, management often asks "Why reinvent the wheel?

Why not use something that has succeeded in the past?" The management fails to ask if it suits the current context.

The intention of quantifying everything is an overhead. Instead of focusing on such overheads, the need of the hour is to understand

the customer's context and their needs and decide what works for them rather than what worked for a different customer two years ago. Customers depend on us to use our technical knowledge, experience and skill to provide them what is best suited for them. Customers don't ask about metrics if you deliver them the best (Unless, you want to overcharge them by showing these metrics). Customer feedback is the right way to measure quality. It's not just counting test cases and defects raised.

I encountered this example of blindly following when I was in need of a new notebook and a pen. The stationary is available at the reception desk in our office where we need to make an entry in the register when we collect the items. I went to reception to get my stationary where the receptionist gave me the register to make the entry. There were already many entries made on the page, I was almost at the end of the page, so I checked the last entry to know what details to enter, it had the following details; Name, Employee ID, Particulars, Quantity, a signature and again the same signature in subsequent column.

I was wondering why he has signed it twice. I then looked an entry above and that too had two signatures. So I scanned a few others entries, all had the same patterns. Now I looked at the column headers. They were as follows; Employee Name, Employee ID, Particulars, Quantity, Employee Signature and Comments. It seems the first person who entered the particulars on that page added a second signature in the comments column by error (or intent?). The twenty other people who completed stationary requests that day copied without even bothering to check why they needed to sign twice.

So what happens if the processes are questioned and analysed? It leads to the discovery of new information which can be raised with the "right" people. It leads to new ways of working. It leads to change. It leads to new ideas. It leads to bugs, issues and further questions. It breaks down assumptions and leads to real improvement.

Process Is Mere Process

Process doesn't find problems, people do. Don't follow the sheep in front of you blindly jumping obstacles that have since been removed, or didn't even exist in the first place. Question, challenge and analyse.

About Author

"Pragmatist, Sapient Software tester, Automation Checker, Practising Context driven testing, Strongly believes Questioning leads to Progress, Hakuna matata.Take life as it comes and make it beautiful,blog at: http://bit.ly/bXUt89 and tweets @sdhanasekar. Favorite quote "Life begins at the end of your comfort zone."– Neale Donald Walsch"

Future Testers In The Making Or Breaking?

BY KRISHNAVENI

This incident took place when I went to attend an interview for a prestigious IT company. While I was waiting for my interview, I overheard a conversation between two other candidates who were seated alongside me. They were talking loudly enough to be heard across the hallway, so I didn't have to strain to listen to their conversation.

One of the candidates started discussing the testing life cycle. I got interested and focused my attention. What I heard was a shocker. Without fearing bad manners, I politely volunteered to correct her.

I told her that what she was saying didn't fit with my understanding of the testing life cycle, so I proceeded to give her what I believe to be the right information. At first she refused to accept my version. I explained where I disagreed with her arguments. She was surprised but accepting that her view of the testing lifecycle could well be wrong. I talked about mine, but I didn't stop there. I wanted to know from where she had studied and learned about testing. What she told gave me the shock of my life.

She had enrolled and paid a hefty sum to a testing institute who sent her a heap of training material to learn remotely. Now, that left me perplexed. I wondered how much of what she learned was right and how much was wrong.

Masters Of The Field

It's shocking to me that companies whose business is to teach testing and who claim to be masters in the field impart misguided and potentially wrong information to candidates. How can a person, totally new to the testing field, ever figure out if they are getting honest, real and accurate information?

Cash Cow Mentality

One of the major problems is that anything that becomes a trend or the "in-thing" and is widely in demand, becomes a very good business opportunity. There are many people who bank on it to make quick money, just as the saying goes "Make hay while the sun shines". Of course that's not wrong, but shouldn't they be providing value for the money they charge? Shouldn't they be imparting the right knowledge, rather than teaching misplaced information without really caring about the aftermath?

Impact On The Testing Community

What is the the impact on the testing community at large? What would be the impact on the companies these people end up working for? Will we get skilled people at work? What kind of quality deliverable can we expect out of people who have been trained this way?

How To Handle This Situation?

It would be good if people, who enrol in training institutes, verify the correctness of the information given to them via trusted, impartial, online resources or by talking to their peers. Unfortunately, this requires a testing mindset, something these people are maybe trying to build upon in the first place.

Read More Testing Books

Read, read and read more. There are many great books on testing avail-

Thanks to Stephen Hill (@Stephen_J_Hill) and Thomas Ponnet (@ThomasPonnet) for volunteering time to help out with The Testing Planet

able. The Internet is also a great source of information to guide new apprentices to the world of software testing. You can join many online discussion forums to share your knowledge and find lots of information for you to browse through.

Network With Passionate Testers

The Testing field has progressed rapidly over the last few years. There is loads of information on testing available on forums and websites including groups of passionate testers who are dedicated to helping grow and steer the testing community.

It's really good to see that passionate testers are willing to share their precious knowledge and are willing to help people learn more about testing.

Find A Mentor For Yourself

Having a mentor makes learning easier by setting a suitable direction and context to your learning goals with someone highly skilled and motivating. It is good to find like- minded testers willing to help budding testers to learn and practice testing.

Search To Evaluate The Companies Credibility

Know more about the institutes before joining in blindly. Research sufficiently. Talk to students who have studied there in the past. Talk to the lecturers and teachers to get an understanding of their personalities, passion and course material.

Ray Of Hope

I hope this has been a good, if not brief, eye opener for the many aspiring people who are thinking about parting with large sums of money and their precious time in the pursuit of becoming a more experienced software tester.

About Author

Name: Krishnaveni
Job title: Senior QA Engineer
About Me: Have been in testing for the past 5 years and am truly passionate about it.

STC Chat

HOSTED BY ROB LAMBERT

Is The Explosion In Blogs About Testing A Positive Thing, Even If We Don't Agree With The Message?

Elizabeth Fiennes: Yes it is as it promotes discussion
Markus Gärtner: yes
Marlena Compton: Visibility is good.
trisherino: yes absolutely!
trisherino: Free flow of ideas. If people don't agree with the message, it will at least get them talking.
Rob Lambert: for sure
Marlena Compton: +1 for that
Markus Gärtner: it helps in bouncing ideas around, thereby making our profession even more professional
Elizabeth Fiennes: and understanding the views of someone you do not agree with help with engaging with them
Markus Gärtner: ok, got a topic for the weekend
Elizabeth Fiennes: which is a great skill in testing
trisherino: Elizabeth: good point, that's a good testing skill
Marlena Compton: There's an American saying, "I disagree with what you say, but will die defending your right to say it."
Elizabeth Fiennes: exactly
Elizabeth Fiennes: except with that particular author, I might just take a Chinese burn on their behalf but not much more
Elizabeth Fiennes:
Markus Gärtner: so, to wrap up on self-education in testing is worthy, maybe we need a different teaching model, besides universities...and certification is just paper
Elizabeth Fiennes: that's the masters you get after the degree from the university of life?
Markus Gärtner: blogging helps to have ideas floating around, and improves our argumentation skills, argumentation skills? Or discussion skills?
Markus Gärtner: oh, and conferences are great to connect and get new ideas
trisherino: It forces you to form ideas properly into words, which I think helps with learning.
Rob Lambert: lol we're off again
Elizabeth Fiennes: Engagement skills
Rob Lambert: engagement skills - interesting.
Rob Lambert: Takes it to the level of a two way conversation with empathy on each side
Marlena Compton: We need much more engagement skills and less debate skills in testing.
Markus Gärtner: +1 Marlena
trisherino: @Marlena completely agree
trisherino: Written engagement skills even. Which can be difficult.
Elizabeth Fiennes: well, 20-ish years ago, we all started off on IRC and it was mental the flame wars that took place
Elizabeth Fiennes: then we graduated to nicer message boards, webpages, IM chats, forums and blogs
Elizabeth Fiennes: and the same flaming is still going on by newbies to the formats
Elizabeth Fiennes: whereas us old hats know that is not the way to engage
Rob Lambert: oh how there have been some flame wars in testing
Rob Lambert: what is the best way to engage?
Marlena Compton: so....
Rob Lambert: when faced with distance
Elizabeth Fiennes: stepping back
Elizabeth Fiennes: asking yourself, is this someone looking for a flamed response, is it worth my time to give it, will me ripping his head off (metaphorically) in a public forum achieve anything?
Elizabeth Fiennes: and if the answers are yes, no and no, then go and find something else to read that is not going to make me go all 1995-IRC on their post
trisherino: I think when it comes to online debates, we have the advantage of being able to take the time to calm down before writing a response, unlike in some face to face scenarios.
Rob Lambert: @trish - indeed
Marlena Compton: twitter, on the other hand.
Rob Lambert: @trish - in our daily jobs it can also be prudent to take a step back and think about things. Especially in heated meetings
trisherino: @Rob That's true as well, if possible.
Elizabeth Fiennes: thing is, I think that slowing yourself down online teaches you the value of holding back your reactionary self and once your embrace that, you can apply it in real life (without realising it) as well
Thomas Ponnet: A couple of thoughts from me after reading the quite substantial discussion. I didn't read it with a glass of wine as Elizabeth suggested, I didn't think that would improve the reply..

Coverage must apply to performance tests, too. Showing the truth (low % coverage) will drive innovation in your testing
#dttip by @ bradjohnsonsv

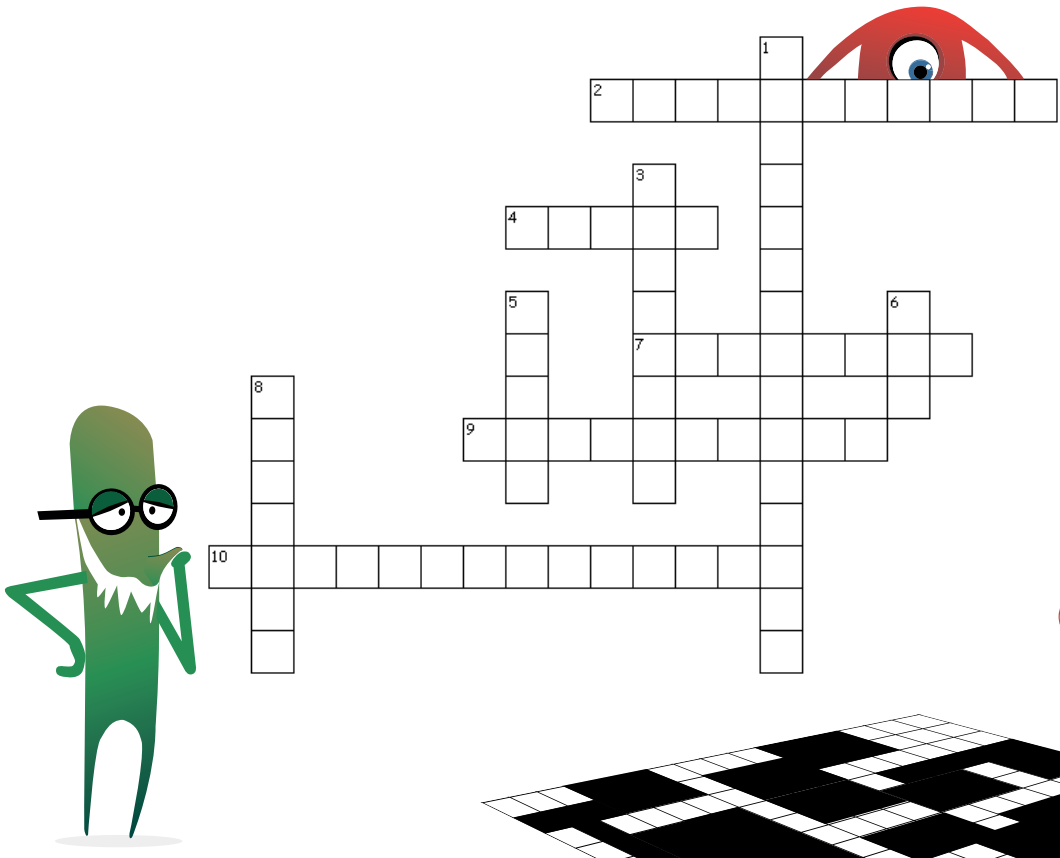
Is Attending Conferences Valuable?

Markus Gärtner: depends
Elizabeth Fiennes: oooh, there is a question
Marlena Compton: yes...it gets you out of your own fishbowl
Elizabeth Fiennes: attending can be expensive, whatever about valuable
Marlena Compton: Especially if you are a solo tester or on a small team.
Markus Gärtner: attending a conference on the solar system may be of lesser value than attending eurostar
trisherino: I've only ever been to STANZ for testing conferences. Went to some developer ones though.
trisherino: but STANZ was invaluable, especially as I was the only tester on my team at the time
Marlena Compton: I went to GTAC and it opened my eyes
Markus Gärtner: I think the professional exchange during the breaks is the best benefit you can get from a conference with a bad program
Marlena Compton: and GTAC is free
Markus Gärtner: oh, it is?
Marlena Compton: yep
trisherino: I found the conference really opened up a whole new world for me, of people who are interested in testing.
Markus Gärtner: you should have told me this one year earlier
Marlena Compton: @trisherino I think I had a similar feeling at GTAC
trisherino: where is GTAC?
Marlena Compton: This year it's at Hyderabad. I attended in Seattle
Rob Lambert: I find them inspiring. Gives me the ideas to try out and the enthusiasm to do so
Markus Gärtner: I noticed that a community of great people also helps my own thinking
Markus Gärtner: it's infectious
Elizabeth Fiennes: you know what I would find really useful as a tester (esp. one working as the dogsbody for a startup) - an online calender of testing events, confs etc
Rob Lambert:check out the SoftwareTestingClub calendar
trisherino: Elizabeth - yes! I have been looking for some calendar like that
Markus Gärtner: last year, GTAC was in Switzerland, I think
Rob Lambert: @markus - the conversations outside of the event are the really great value add.
Elizabeth Fiennes: I tend to find out about confs in retrospect when ppl blog about them
trisherino: There never seems to be many testing events happening in Australia. But maybe I'm just not aware?
Marlena Compton: @trisherino You are right.
Rob Lambert: Start one
Marlena Compton: I've counted 3
trisherino: Marlena: good time for us to start some?
Rob Lambert: yay
Marlena Compton: A peer conference or unconference might be fun.
Rob Lambert: open space events are fun
Markus Gärtner: started to love the open space parts at the last few confs I visited
Marlena Compton: New Zealand is having one soon. Wish I could go.
trisherino: Marlena: I'd be happy to help organise something in Sydney sometime.
Marlena Compton: @trisherino We can talk about it.

So Is A Diploma With Work Experience A Way To Go?

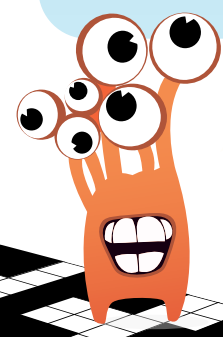
Elizabeth Fiennes: For me, yes
Markus Gärtner: that depends
trisherino: Sounds like a good idea.
Marlena Compton: Seems like it assumes certain limits to software.
Markus Gärtner: diversity in work experience would be more interesting to an employer, I think
trisherino: I guess it depends on what's being taught?
Marlena Compton: I think I agree with what you said.
Rob Lambert: I have worked in many industries and each one has needed a new set of ideas and principles not to mention new people with different skill sets and new terminology
Marlena Compton: Do you mean "testing diploma" or "a diploma"
Markus Gärtner: well, I think it's unlikely that you learn everything at a single company, and that everything you learned there is of any meaning at a different company
Elizabeth Fiennes: I mean a software testing diploma
Marlena Compton: that assumes limits to software
trisherino: It would be nice to have something like that to give wannabe testers a starting point to further their own education through self-learning and experience.
Rob Lambert: @markus - absolutely. Staying at the same company can often be detrimental
Rob Lambert: you need to explore how different places work
Elizabeth Fiennes: That's true Rob but in every company I have gone to, I have drawn on skills that I have learned somewhere else
Elizabeth Fiennes: (and developed some new ones)
Rob Lambert: but have these been general skills suitable for any industry
Markus Gärtner: (weekend testing can provide you with some experience in different applications, if you like to take the challenge)
Elizabeth Fiennes: Yes, the ability to think quickly, stay calm, not throw heavy objects at PMs, all very transferable skills
Rob Lambert: commuinication? Analysis?
Rob Lambert: Social skills? Learning? Self teaching?
Rob Lambert: Being unafraid to ask questions?
Marlena Compton: So would the same "testing diploma" prepare me to test at Microsoft and Atlassian?
Markus Gärtner: i think there is a difference between experience in terms of practically applying something, and knowing about it
Markus Gärtner: the most critical thing for myself was my ability and willingness to learn new things on my own
Markus Gärtner: so, learning is a worthwhile thing to have
Rob Lambert: @markus essential some might say?
trisherino: Sounds about as good as any other diploma / degree - enough to learn some terms and concepts, but then you get into the industry and realise you don't know a damn thing.
Rob Lambert: @trish - good point. I've seen that first hand
Markus Gärtner: I learned some of that during my university courses (to get back to Rob's question from earlier)
Marlena Compton: I stick by my Interdisciplinary Studies degree. The CS degree has made it easier, but I could have done without it.
Markus Gärtner: I don't know whom to attribute it, but "I'll stop learning when I'm dead."
Rob Lambert: If you think you've reached the top you are already on your way down
Elizabeth Fiennes: There's a top to software testing?
Marlena Compton: yeah....less specialization
Elizabeth Fiennes: I heard it as I'll stop improving when I'm dead but same principle

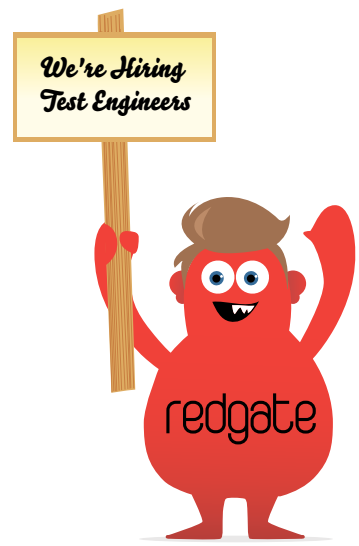
Weather & Games

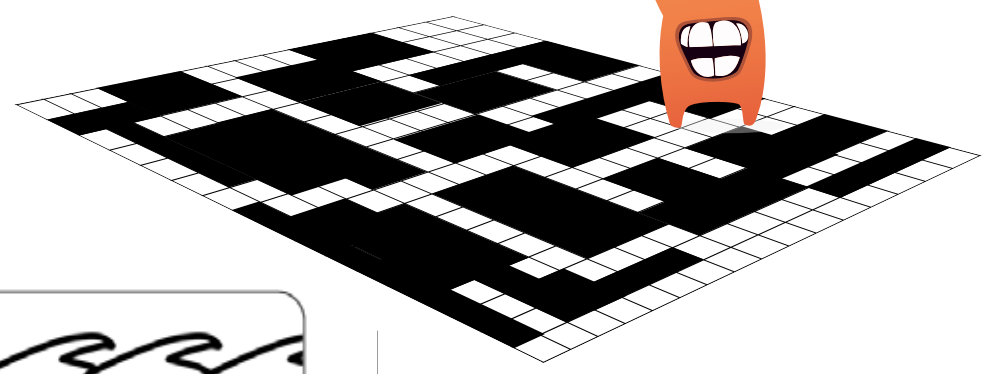


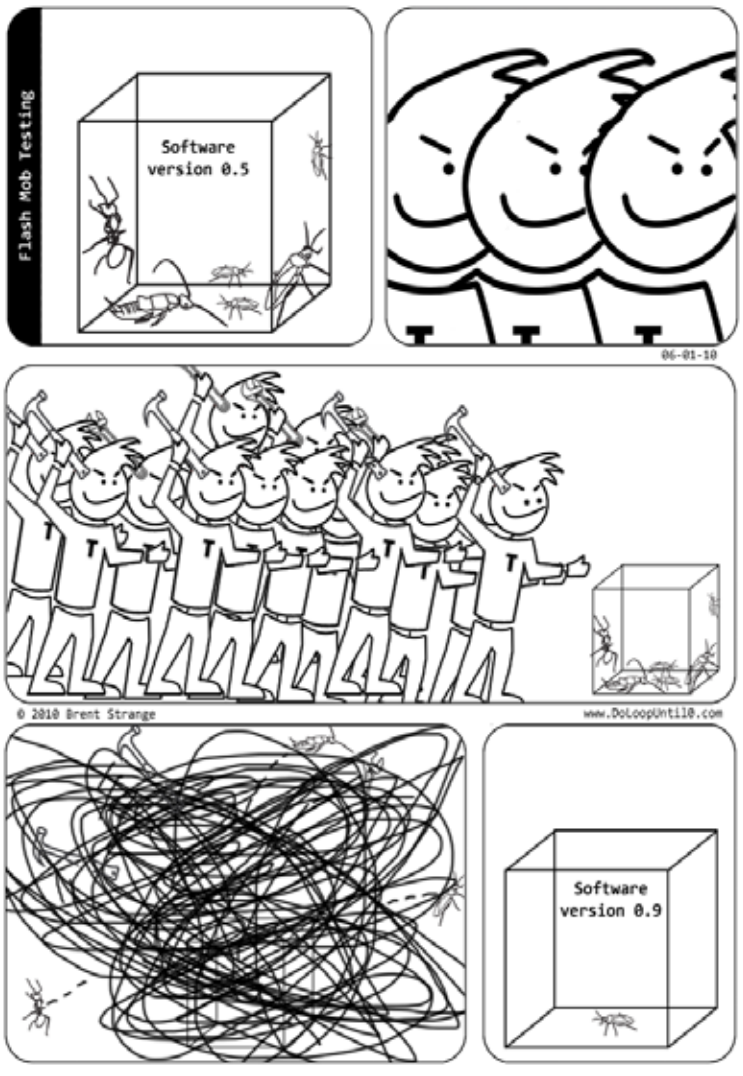
Across
2. Simultaneous learning, test design and test execution
4. An iterative, incremental framework for project management and agile software development
7. A name of a test tool and a chemical element
9. The real name of 'The Social Tester'
10. A software implementation of a machine

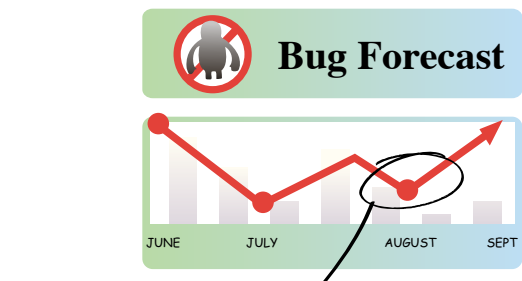
Down
1. A type of people who join The Software Testing Club
3. A name of a test conference that shares it's name with a train service
5. An open source automated acceptance testing and ATDD framework
6. Also known as an error
8. A type of team also confused with a test team











See what lovely @testingclub services are on offer - http://bit.ly/d88e69