softwaretesting Club

a community magazine

Oh Hello There

And welcome to the new Software Testing Club Magazine.

It's very exciting to be kicking off 2010 with the release of The Software Testing Club magazine. It's a brand spanking new magazine full of articles, news, fun and games. In keeping with the vibe of the STC we've kept the magazine fresh, fun and focused on you, the testers.

This premier edition sees tall tales of new code, stories of agile success, advice on gatekeeping, real life ghost stories, blog updates, cartoons and much more.

Think of this magazine as your little bit of fun. Your source of testing stories. Your little bit of downtime. Your little magazine of testing. And feel free to share this magazine with your friends and colleagues. Ping it around and share the fun.

There's no doubt 2010 is going to be an exciting year for software testing. With some great initiatives and concepts forming and with technology moving at such a rapid rate, it's going to be a challenging yet interesting year for many in our profession.

For some testers it's early days, they're learning the craft and building their skills, getting to grips with what testing is, finding out more about the craft and gaining valuable work experience. For others this year is about fine tuning their skills, building their learning, promoting testing and spreading the word. No matter what your testing goal is for 2010 though let the Software Testing Club be there to help you along.

We are on the lookout for more articles, stories and much more for the next edition so please get them sent in to rob@softwaretestingclub.com.

The main peeps behind the magazine:

Rob Lambert
The Social Tester
http://thesocialtester.posterous.com/

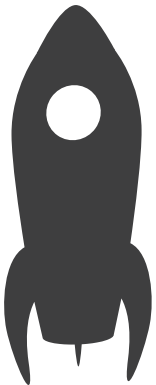
Rosie Sherry
The Social Creator
http://rosiesherry.com/

Joel Montvelisky
The Anti-E Social Tester
http://qablog.practitest.com/

Phil Kirkham
The Social Builder
http://expectedresults.blogspot.com/

A huge thanks and high-fives go to all who contributed content and all who worked behind the scenes proof reading and helping out to make this magazine possible.

Contents



Carnival of Testers
Activity from the blogosphere

The blog-activity-ometer has been very busy in the last 3-4 months. Plenty of output, plenty of discussion, topic cut'n'thrust and the odd (sometimes very odd) amusing post.

Let's take a sample from the smorgasbord of delights that have been those blog posts.

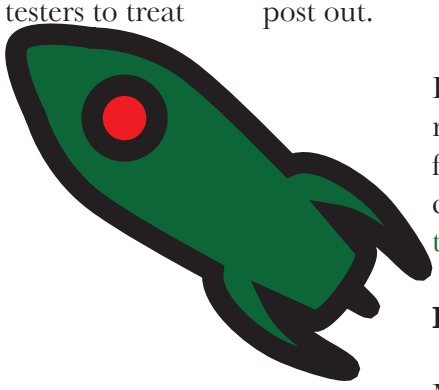
Testing vs. Checking

Let's start at the high-end of the Richter scale... Michael Bolton started writing about the "testing vs. checking" subject in a number of posts (the first one here). This was a topic that generated a lot of discussion, comments (on various sites) and blog posts - including approval, disapproval and I was waiting for the Spanish Inquisition (Monty-python style of course) but I think Michael has escaped the torture of the "comfy chair" so far.

Several people wrote (or tested and explored their thoughts) on it and aspects relating to it, including Albert Gareev, Martin Jansson, Simon Morley, Ben Simo and Trish Khoo.

Fair Play

Articles were written encouraging testers to treat each other with courtesy, politeness, fairness, respect and to not pre-judge. People stepping up to encourage this were Lannette Creamer, Lisa Crispin, Catherine Powell, Matt Heusser and Jon Bach. Worth bearing in mind when the discussions are getting heated...



Weekend Testers

Probably the most recent new group has been the weekend testers. Many different people have written enthusiastically about the idea. Check out James Bach's post, Bharath's summary and Parimala Shankaraiah's experience report.

New Faces?

The last few months of 2009 saw a bunch of testers starting to blog.

Yvette Francino bounced onto the blogging scene in September. She has been writing about a range of topics and issues connected with her learning of different test-related areas. In this post she reviews the progress made since starting Beyond Certification.

The end of September saw Peter step onto the blogging scene. He covers a wide range of triggers and influences in his posts, this look at learning styles and philosophy was a typical example.

November saw Dave Whalen land on the blog-o-scene with a look at bug report priority and severity.

Seth Eliot started writing in December and has produced a few posts worth checking out. One of his latest was a post covering the code coverage discussion between Matt Heusser, Alan Page and BJ Rollison. For an enlightening view of the discussion check his "exciting code coverage" post out.

Rounding off the year of new and recent bloggers John Stevenson put finger to keyboard. Another tester's critical eye being cast on issues of the day.

Random Pick?

Misko Hevery provided an intro to getting started with your own TDD. Get past those nightmares about TDD!

Anne-Marie Charrett helps you discover if you're incompetent without knowing about it! Don't have nightmares if you know that you don't know something!

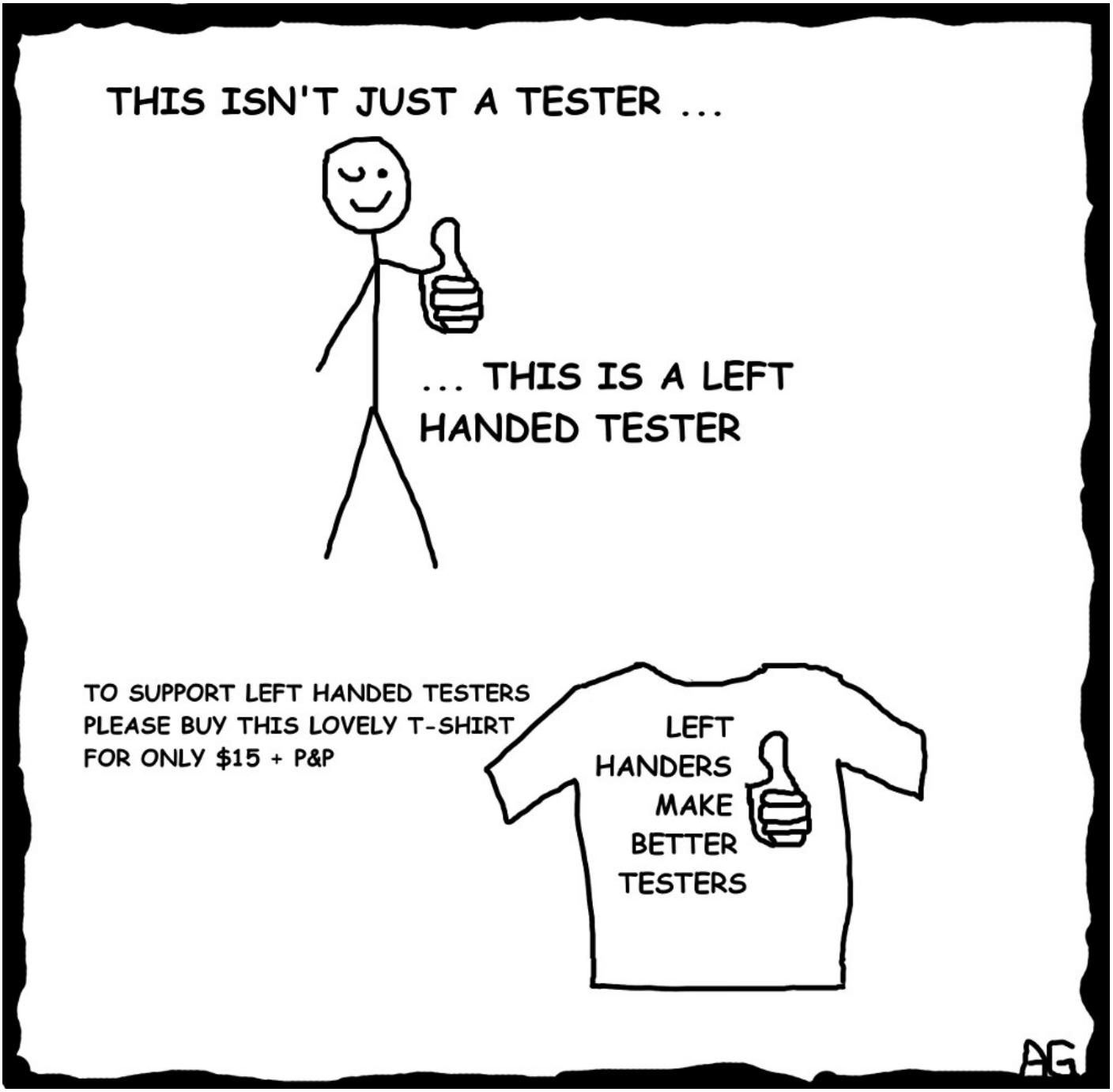
Yours-truly on ways of tackling software testing myths by relating them to absurdity. Don't have nightmares just "strange dreams"!

Adam Goucher's post on great public speaking tips. Great tips about great speaking! No need for nightmares about public speaking!

Markus Gärtner's post and photo collection at his software craftsmanship Pecha Kucha. Don't have nightmares about the chicken!

Rob Lambert's encounters with the animal kingdom and planning. Yes, time to have nightmares about the rabbit!

by Simon Morley. He blogs at: http://testers-headache.blogspot.com/



By Andy Glover: http://cartoontester.blogspot.com/

Motivating staff and traditional techniques

by James Christie

Recently I've been thinking and writing about the effects of testing standards. The more I thought, the more convinced I became that standards, or any rigid processes, can damage the morale, and even the professionalism, of IT professionals if they are not applied wisely.

The problem is that calling them "standards" implies that they are mandatory and should be applied in all cases. The word should be reserved for situations where compliance is essential, eg security, good housekeeping or safety critical applications.

I once worked for a large insurance company as an IT auditor in Group Audit. I was approached by Information Services. Would I consider moving to lead a team developing new management information (MI) applications? It sounded interesting, so I said yes.

On my first day in the new role I asked my new manager what I had to do. He stunned me when he said. "You tell me. I'll give you the contact details for your users. Go and see them. They're next in line to get an MI application. See what they need, then work out how you're going to deliver it. Speak to other people to see how they've done it, but it's up to you".
The company did have standards and processes, but they weren't rigid and they weren't very useful in the esoteric world of insurance MI, so we were able to pick and choose how we developed applications.

My users were desperate for a better understanding of their portfolio; what was profitable, and what was unprofitable. I had no trouble getting a manager and a senior statistician to set aside two days to brief me and my assistant. There was just us, a flip chart, and gallons of coffee as they talked us through the market they were competing in, the problems they faced and their need for better information from the underwriting and claims applications with which they did business.

I realised that it was going to be a pig of a job to give them what they needed. It would take several months. However, I could give them about a quarter of what they needed in short order. So we knocked up a quick disposable application in a couple of weeks that delighted them, and then got to work on the really tricky stuff.

The source systems proved to be riddled with errors and poor quality data, so it took longer than expected. However, we'd got the users on our side by giving them something quickly, so they were patient.

It took so long to get phase 1 of the application working to acceptable tolerances that I decided to scrap phase 2, which was nearly fully coded, and rejig the design of the first part so that it could do the full job on its own. That option had been ruled out at the start because there seemed to be insurmountable performance problems.

Our experience with testing had shown that we could make the application run much faster than we'd thought possible, but that the fine tuning of the code to produce accurate MI was a nightmare. It therefore made sense to clone jobs and programs wholesale to extend the first phase and forget about trying to hack the phase 2 code into shape.

The important point is that I was allowed to take a decision that meant binning several hundred hours of coding effort and utterly transforming a design that had been signed off.

I took the decision during a trip to the dentist, discussed it with my assistant on my return, sold the idea to the users and only then did I present my management with a fait accompli. They had no problems with it. They trusted my judgement, and I was taking the users along with me.

The world changed and an outsourcing deal meant I was working for a big supplier, with development being driven by formal processes, rigid standards and contracts. This wasn't all bad. It did give developers some protection from the sort of unreasonable pressure that could be brought to bear when relationships were less formal. However, it did mean that I never again had the same freedom to use my own initiative and judgement.
The bottom line was that it could be better to do the wrong thing for the corporately correct reason, than to do the right thing the "wrong" way. By "better" I mean better for our careers, and not better for the customers.

Ultimately that is soul destroying. What really gets teams fired up is when developers, testers and users all see themselves as being on the same side, determined to produce a great product.

Reality is chaotic. Processes are perfectly repeatable only if one pretends that reality is neat, orderly and predictable. The result is strain, tension and developers ordered to do the wrong things for the "right" reasons, to follow the processes mandated by standards and by the contract.

Instead of making developers more "professional" it has exactly the opposite effect. It reduces them to the level of, well, second hand car salesmen, knocking out old cars with no warranty. It's hardly a crime, but it doesn't get me excited.

Development and testing become drudgery.

Handling the users isn't a matter of building lasting relationships with fellow professionals. It's a matter of "managing the stakeholders", being diplomatic with them rather than candid, and if all else fails telling them "to read the ******* contract".

This isn't a rant about contractual development. Contracts don't have to be written so that the development team is in a strait-jacket. It's just that traditional techniques fit much more neatly with contracts than agile, or any iterative approach. Procurement is much simpler if you pretend that traditional, linear techniques are best practice; if you pretend that software development is like civil engineering, and developing an application is like building a bridge.

Development and testing are really not like that all. The actual words used should be a good clue. Development is not the same as construction. Construction is what you do when you've developed an idea to the point where it can be manufactured, or built.

Traditional techniques were based on that fundamental flaw; the belief that development was engineering, and that repeatable success required greater formality, more tightly defined processes and standards, and less freedom for developers.

Good development is a matter of investigation, experimentation and exploration. It's about looking at the possibilities, and evaluating successive versions. It's not about plodding through a process document.

Different customers, different users and different problems will require different approaches. These various approaches are not radically different from each other, but they are more varied than is allowed for by rigid and formal processes. Any organisation that requires development teams to adhere to these processes, rather than make their own judgements based on their experience and their users' needs, frequently requires the developers to do the wrong things.

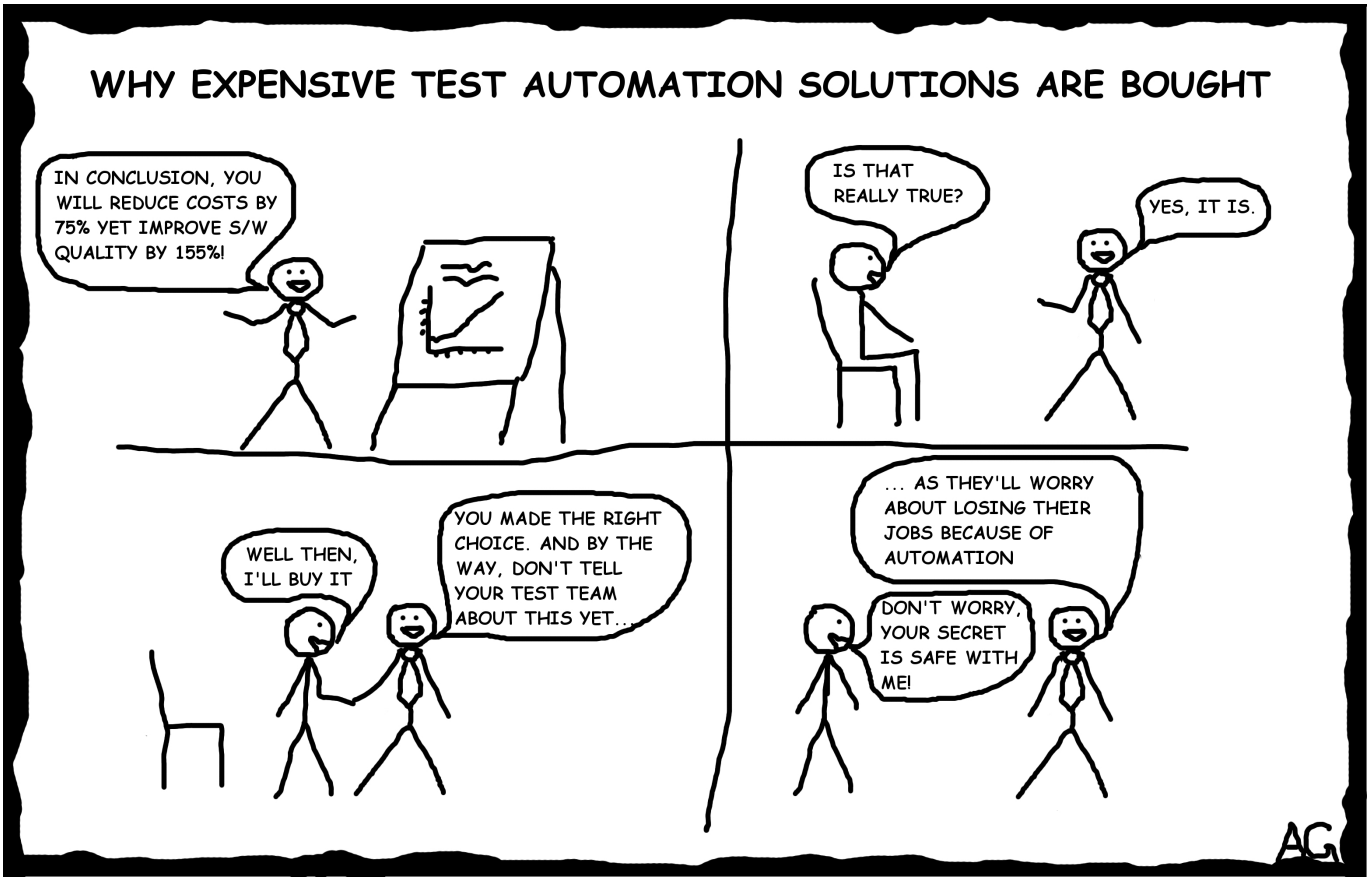
This is demoralising, and developers working under these constraints have the initiative, enthusiasm and intellectual energy squeezed out of them. As they build their careers in such an atmosphere they become corporate bureaucrats. They rise to become not development managers, but managers of the development process; not test managers, but managers of the test process. Their productivity is measured in meetings and reports. Sometimes the end product seems a by-product of the real business; doing the process.

If people are to realise their potential they need managers who will do as mine did; who will say, "here is your problem, tell me how you're going to solve it". We need guidance from processes and standards in the same way as we need guidance from more experienced practitioners, but they should be suggestions of possible approaches so that teams don't flounder around in confused ignorance, don't try to re-invent the wheel, and don't blunder into swamps that have consumed previous projects.

If development is exploration it is thrilling and brings out the best in us. People rise to the challenge, learn and grow. They want to do it again and get better at it. If development means plodding through a process document it is a grind.

I know which way has inspired me, which way has given users applications I've been proud of. It wasn't the formal way. It wasn't the traditional way. Set the developers free!

James Christie is an independent software testing consultant.
www.clarotesting.com



By Andy Glover: http://cartoontester.blogspot.com/

I'M NOT JUST A TESTER I'M MORE THAN THAT

BY PETER HAWORTH-LANGFORD

Experience can shape you forever right?

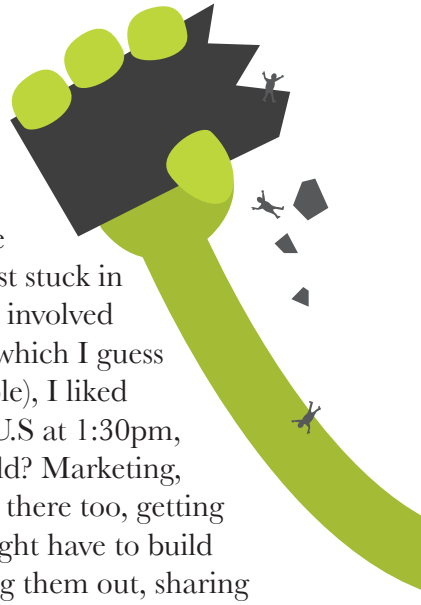
I remember one of the first jobs I had was in the airline industry where data and data accuracy is of supreme importance. I joined when the company had started branching out into electronic products, that was a great coincidence for me. One of the things I was involved in was producing shrink wrapped software for large corporations. There was a big mainframe that took feeds from all the major airlines to produce a massive database. There were cobol/jcl jobs, that ran with lots of rules, I mean loads…they ran overnight or over a weekend and produced an extract. That extract had to be 'checked' and then got downloaded to a PC. Batch jobs then converted the data into a form that could be presented in a commercial application.

So what was my involvement? Well I did some extract 'checking'. Well what I call checking, I ran the batch files, checked the output from those, built the software, installed it, tested it with the new data. I then Burned a gold CD or 'master' floppy disks (trust me putting data onto 15 floppy disks was laborious but it must have been worse for the customer who had to install them… we quickly moved to producing CDs only), and then to 'production'. I really loved that job! I got to talk to the mainframe guys (did everything run as expected?) It was all about abends! Does anybody remember those? The batch file and software developers (they were mostly based in the U.S. and I was based in the UK).

Any batch file processing problems? Well if I found them in the morning I would have to wait until 1:30pm to give the developers a call. I loved those guys! (If I ever got the chance of putting a dream team together they would be in it!). Once I'd done my stuff and got a 'master' it was off to production, they would take my CDs /floppies send them off to a 'replicator' who would take the masters and produce 1000's of copies.

Who else would I work with? Well I used to work closely with marketing (they were generally interested in how things were going, we'd make specialized products for customers, which 'marketing' generally controlled along with in-product advertising) .

Anyone else? The helpdesk, I worked with the helpdesk sharing product knowledge, trying out customer issues on the products (because I was probably the only person with every latest product version, and an environment to install on if required) .



So why am I sharing this with you? You've heard this all before haven't you?

I don't want to just execute tests, I don't want to be just stuck in testers corner. I want to be involved with the mainframe guys (which I guess are now the database people), I liked talking to the Devs in the U.S at 1:30pm, it involved me in their world? Marketing, yes I liked the involvement there too, getting an insight into what we might have to build next. The helpdesk, helping them out, sharing my knowledge, questions like 'We've got a customer who did x? can you check this out?' (They never said 'can you test this?')

I still remember a call to this day about a customer who said 'We gotta virus! The cursor keeps falling off the bottom of the screen'.

After a management panic we worked it out in the end…the customers lever arch folder was stuck on the enter key…The weight of the folder on the enter key on the keyboard kept it depressed which would make the cursor move and not allow you to do anything else.

In another job, I was asked to produce the accessibility guide for developers, there was absolutely no test execution involved in this activity, it got published and as far as I know the developers followed it. Was that a good thing? I thought so, but more importantly I got to learn all about web accessibility.

Just a tester?...maybe

More recently I've been more closely involved with customers, sometimes I feel customers are not sure what they want, that's fine that's their prerogative right? Sometimes they can have a vision but finer details require 'walking through', someone who is the between the techies and the non-techies.

Just a tester?…maybe

What about using 'sapient processes to question a product in order to evaluate it' or

'Making an observation linked to a decision rule that may be performed by a machine…'

Is this a tester?…maybe

Have you ever worked with a 3rd party and you had to do some testing? Well the 3rd party are providing you a service aren't they? In this situation you could be the customer couldn't you?

What about creating some tools to help with your testing? You are also the toolsmith aren't you?

You have knowledge of a particular area of the system? You could be the domain expert couldn't you?

We do management as well don't we? Testers doing management? Yes. We organize things we want to test, things around what we want to test (test environments for example) all to timescales. That's time management isn't it? What about defect lifecycles? That's defect management isn't it?

Well I want to be involved in everything, everywhere. Everyone could do with an extra person couldn't they? Does that fit into the definition of tester? How about 'a bit of everythinger'? A 'Floaty' - A floating person? The chewing gum in-between everything? Yeah I'll take that. A solver? Yes I like that I like trying to solve things. Or maybe the workeroutta? Requirements, Designs, applications…

How about all the learning, practicing and improving I want to do?

Just a tester?…

How about part of the team?

By Peter Haworth-Langford
http://007unlicensedtotest.blogspot.com

The Ghost

by Luisa Baldaia

I would like to tell you a story. A story about the time a Software House starts testing its own software.

You might think it strange, start testing? What did they do before? It's not that strange. In Portugal many software houses still don't do formal testing, at least the way we think about testing.

In this company they started testing after a reorganisation of teams. They moved from teams specialized in different business areas, to teams specialized in different functional areas. Everyone was distributed in and around the most suitable new team.

Before the reorganisation (the new development model) each person on her team was analyst, programmer, tester and documenter, all at the same time. All the roles were there but distributed in another way. The results weren't bad but the number of code lines written was not that large either.

After the reorganisation the one team who experienced the most difficulties in adapting to the new role was the Test team. Not that the role was completely new to them, but it was the first time that the only task they were asked to do was Testing. They were a small team too.

> "It was so simple, why didn't we think about it before? And we can develop more like this to apply to other processes"

In the past a decision was taken in the Automation direction. They bought an expensive tool, one of the best in the market, and started working. Should I say working? No. They tried to put it to work but they didn't prepare themselves properly. It was necessary to identify the right features, functions and processes to automate, define test cases, create test data, evaluate maintenance cost, etc. Unfortunately, this homework and prep work was not done and the result was a bad experience and a bad investment. The first step was itself the first error, buying a tool without identifying the needs.

Back to the new (reorganisation) era, how do we test now? I could say we practice a little of Exploratory Testing and a little of Checking. We work in a month development life cycle. A lot of new procedures were defined in Quality, like Acceptance Criteria (the point when it is acceptable by the client), Regression Tests and Maintenance tests.

Two years have since passed and little innovation has been seen in the testing process. Yes, we have become more experienced. Yes, we now have bug tracking tools. No, we didn't achieve as much Quality as wished. Yes, we started to think about some Automation again.

We had avoided Automation. The ghost of the past was still there, and he wasn't helping our Team.

Even so we started to think about Automation again. We needed to.

One day, one of the testers faced a problem. There was a critical high use process that was going to be modified. The changes were deeply affecting, so the project manager decided to do it in 3 phases. The process, that was about to be modified, normally processes a large amount of data and it was mission critical. So the tester thought that there was only one way to assure that things would keep working well after each phase: automate tests.

So this tester started to work on it. In a few hours he built some simple scripts. Then he found a way to feed them with existing test data. To finalize he created some batch files and scheduled them to run every night.

Automation was done and it was so easy and quick to do. After a few testing sessions he presented the results and the work to the rest of the team.

What do you think they thought about it?

Every one was astonished. "It was so simple, why didn't we think about it before? And we can develop more like this to apply to other processes".

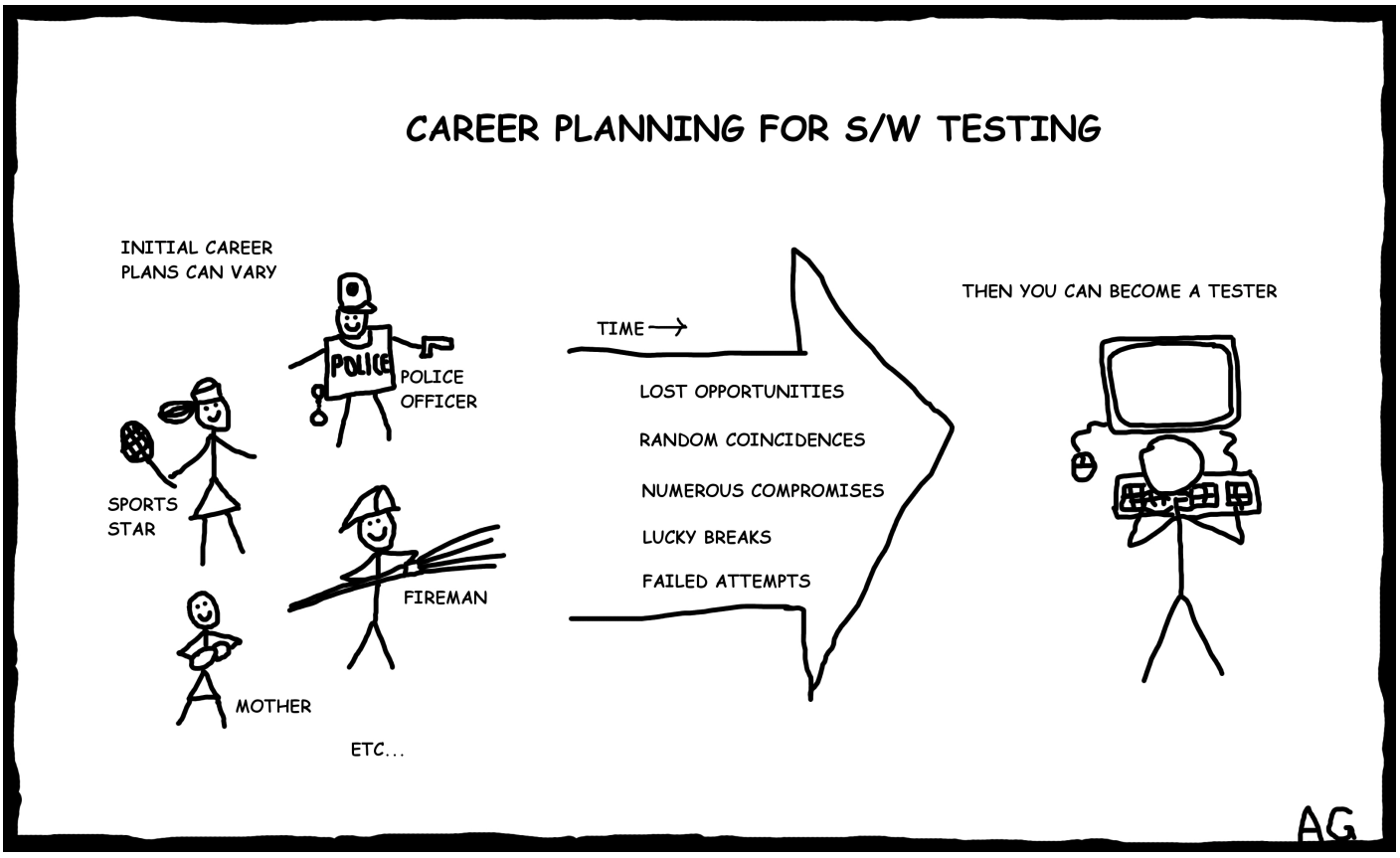
The egg of Columbus - this is what they called it. (http://en.wikipedia.org/wiki/Egg_of_Columbus)

The rest of the story is easy to imagine

The great thing about this story is that a simple idea was capable of killing the Ghost. The Ghost was gone.

In all companies, in all teams, on all projects there are ghosts. It's in our hands to beat them.

By Luisa Baldaia - QA Lead at Primavera Business Software Solutions



By Andy Glover: http://cartoontester.blogspot.com/

THE EMPEROR'S NEW CODE

BY ROBERT HEALEY

There once lived an Emperor of marketing who was so excessively fond of new processes, practices, methodologies and fads that he spent his entire R&D budget on them, just so he could keep one step ahead of the competition. Just as they say of other senior managers that they "keep in mind the balance sheet", so the Emperor was always said to be "reading Business Week".

The company, being new and dynamic, (and without a sign on the door) attracted many strangers; hawkers, traders and purveyors of religious services. One day, two rogues came, they gave themselves as "Dreamweavers – the ultimate developers" and declared they could design, code and implement an entire e-commerce strategy for the company. They announced that their coding skills were the finest that could be imagined. Not only were their interfaces, they said, so uncommonly user-friendly that documentation would be unnecessary but also that their engines were so robust that testing was pointless. Websites generated using their code had such wonderful quality that they became meaningless to anyone who was incorrigibly stupid or was unfit for the office he or she held.

"What a capital idea?!" thought the Emperor. "If I should launch such a site, I should be able to find out what employees are not fit for the positions they have and should be able to spot the easy-picking customers that are dunces. Yes, this code must be compiled for me on all platforms immediately!"

And he gave the rogues a great deal of cash in hand and a plan of iterative development with regular previews so they might begin their work at once.

As for the Dreamweavers, they turned on two netbooks and pretended to be working but they were only reading news-sites (and occasionally softwaretestingclub.com). They demanded the finest commercially licensed code and the costliest machines on which to compile; these they auctioned on e-bay and worked on surfing the 'net till late into the night.

At the end of the first iteration the Emperor sent his Chief Financial Officer to measure the burn-down velocity and see that work was in compliance with ISO 9003 (he'd Googled it). He would have gone himself except that he felt that since those not fit for their offices would fail to derive meaning, it was important at such times to emphasise task delegation, although he definitely had nothing to fear for himself. That, and he had indigestion, a headache and his mother-in-law was coming to visit. But definitely nothing to fear; at all. All the employees knew what peculiar power their promised company site would possess, and all were anxious to see how bad or stupid their colleagues were.

"I will send my honest CFO to the weavers," thought the Emperor. "She can best judge how it looks, for she has sense and no-one understands that Sarbanes-Oxley stuff better than she."

Now the good and hugely-overworked CFO went into the development suite, (with newly installed pool-table, arcade games machines and espresso coffee maker) sat at a computer terminal and opened her eyes wide. "I cannot see anything except this blue screen of death!" But she did not say this.

Both the cowboys begged her to be so good as to come nearer (and to bring two coffees while she was standing) and asked if she approved of the developments in the engine, style sheets and underlying architecture. They then pointed to the top right corner, and the financial officer stared at the screen; but she could see nothing because there was nothing to see.

"Crikey!" she thought, "seven years in business school, can I indeed be so stupid? I never thought that, and the Association of Chartered Accountants must not know it. Am I not fit for a rise to a major industry player and out of this pokey company with no e-commerce strategy? No, it will never do for me to tell that I could not see the stuff."

"Whadya think?" asked one coder as he went on typing in shell scripts.

"Oh, it is remarkable – most robust!" answered the career-focussed financial officer as she peered through her spectacles. "What efficiency of code use; just one line to produce all that! I shall email a report to the Emperor that I am very, very pleased with it."

"We are glad of that," said both Dreamweavers; and they spoke of bashing the finest shells, of using the heart of Python with the most exquisite routines using mother-of-Perl. They said the Java had kept them up for nights on end, so strong was the type used. The CFO recorded all points using shorthand, that she might be able to report it accurately. She did this so.

Now the rogues asked for more money, unregistered software licenses, and stock options on top Fortune 500 companies, which they declared they needed to incentivise their coding motivation. They exercised all in-the-money options and funnelled all proceeds to accounts in the Cayman Islands; not one line of code was typed in return. They continued to surf on social network sites, dubious chat rooms and low-cost airline ticket vendors as before.

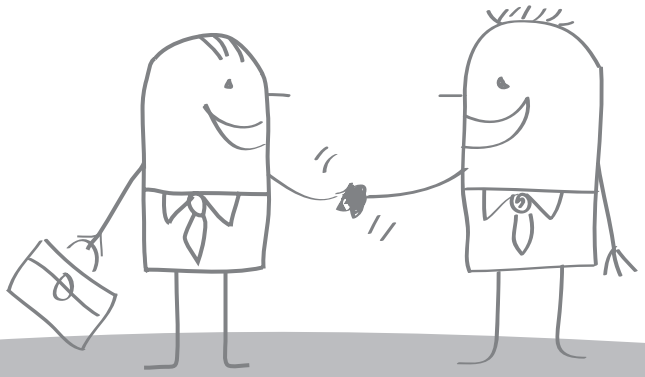
The second iteration end soon approached and the Emperor dispatched another honest officer of the board, the head of Human Resources, to see if the deadline would be achieved. He fared just as his colleague: he looked and looked, but as there was nothing to see except a blue screen with "CAN PIG RIDE". It was totally meaningless to him.

"Have you ever seen a neater bit o' stuff guv'nor?" asked the two rogues who, after watching too many parodies of the one scene from "Downfall" on YouTube, had developed faux London cockney accents. They displayed and discussed the Qt GUI of HTML, XML which was pure Flash. They said it was their unique combination of C, C++ and C# that made the display see-through to those stupid or unfit. It was something about the common PASCAL root and, for some people, the pressure was just too much.

"That last pun was awful!" inwardly groaned the HR chief "but I also know I am unfit to work in HR as I barely passed my English bachelors. I've years of student loans to repay so I must not let it be noticed." And so he praised the stuff he did not understand and expressed his pleasure at the wonderful design and intuitive interface. "You'll love it," he informed the Emperor.

At the office watercooler all the topic of conversation was of this amazing new code. How beautiful it was; how much easier each of their jobs would become; how much more work would result; where a box could be found at short notice in case, you know, I had to pack up my desk or something. At the end of the final stitch-up phase the Emperor had procrastinated enough and signed-off on too many unforeseen expenses, he had to see his new code for himself. With the entire staff in tow (including the co-ops), he went to the two cunning "developers" who were now typing furiously without any text editor being open.

"Isn't that amazing?" said the CFO and head of HR simultaneously. The head of HR then called "jinx", which he later came to regret and compensate for as a result of an equality tribunal. They pointed at the large blue screen with "C.A.N. P.I.G. R.I.D.E." in the centre for they thought that the others could see the new site.



The Emperor who had grown up with texts-peke and never sought to question abbreviations thought, "OMG! I can't see anything. I must be dumber than two short planks. What is 'can pig ride' suppose to mean? I mustn't be fit to be Emperor? In the circumstances it's pretty cool that I am then. Great. LMAO." Aloud he said, "That's tremendous! Great work guys! Well worth the effort. This is revolutionary!" All present ooh'ed and aah'ed as the rogues described all the various features of the site. Anyone who was offered control of the mouse to navigate for themselves politely but firmly refused and claimed that it was so immediately intuitive as to warrant no further action. The staff recommended that the Emperor should launch the site as soon as possible to share its wonders with the world. The whole company seemed to be in general celebratory mode, the Dreamweavers were given immediate performance bonuses (approved by HR and the CFO) and all headed down to the pub for a drink and to wonder privately how best to polish their CV's.

The whole night before the site went live the rogues were awake, playing multiplayer online games against a team in Lesotho, Africa and finally ensuring that all browsing histories and cookies were removed from their machines and that anything small enough and not nailed down was in their travel bags. The Emperor himself had organised a massive viral marketing campaign with A-list celebrities endorsing the wonders of the new company website. The scripts for the 30 second advertisements were written by the developers and included the lines, "See how easy it is to use, how quickly to load", "You'll be surprised by our attention to detail" and "Guaranteed to work 100% of your time". Press releases were issued to all major tv, radio and newspaper networks extolling the benefits and, ultimately, return to shareholders. The public launch was broadcast live from the town square and shown in 14 different countries in six languages and was being podcast later that day.

As the Emperor pressed the eponymous big red button and the sites "homepage" appeared on the giant screen behind him. Everyone in the press corps said, "How incomparable is this site to all others?! What a technological breakthrough?! Truly a paradigm shift in web development!" None let it be perceived that they were stupid or unfit for their salary.

"Would the whole lot of ye ever cop on?" a little Irish child cried out at last. Slowly at first and then with gusto the murmur rumbled through the crowd. First of confusion, then of agreement and then of mirth until at last the whole audience was shaking with laughter and shouting as one, "the Emperor has no cop on!"

Epilogue:

The child grew up to be a software tester, extolling the virtues of early and frequent testing.

The Emperor settled most of his lawsuits with former employees out of court, got a major book deal and sold the rights to have it made into a major motion picture.

The rogue developers used the proceeds of their scam to emigrate to Eastern Europe where they now run a major spam factory.

And they all lived happily ever after.

Agony Aunt

As a bug finding tester I rock. To totally rock I need to work on my soft skills. What's the best way to acquire these?

Well, I guess the first thing that I need to do is to clarify what soft skills are. In my view, the term 'soft skills' (Emotional Intelligence) can be defined as a cluster of personality traits, social culture, communication, language, personal habits, approachability and general outlook (optimism) that characterise the relationships with those around you. As you've already identified, these skills play an important part in your contribution to the success of a project and ultimately to the organisation.

So, when considering your professional career as a tester, which soft skills are more important than others?

Daniel Goleman's best seller Emotional Intelligence: Why it can matter more than IQ, is an interesting read, although the real answer to this question does of course, depend upon your role.

However, to concentrate on your approachability, communication and optimism would prove to be a solid starting point.

What is the best way to acquire these? There are obviously lots of websites, books and training courses that specialise in a host of soft skills, and depending upon your confidence, consider a training course that will enable you to get feedback in a really safe environment. This is particularly important when learning or improving soft skills, as measuring success can be more subjective than, say, passing an exam, or writing a piece of code.

In saying all of that, here is a pragmatic approach that you may want to also consider:

To break the job down into more manageable and achievable sizes, take each skill you want to concentrate on and ask yourself these questions:
Supposition: Everyone has some level of soft skills.

For each skill ask yourself:
1. What do I do now?
2. What happens when I do this?
3. How would I like this to be different?
4. What, specifically, would I need to do differently? (Try to state this in the positive; I need to do……or more of…… (rather than stop doing, or do less of etc.).
5. Of everyone I know, who does this well?

Once you have found a person for the answer to #5 take the following actions :

What specifically do they do (notice their behaviour, what and when they communicate; their body language, how and when they do this…)

Visualise or if it's easier, talk yourself through you doing these things.

What do you hear yourself say? (Pace, tone, the language itself etc.)

How do you feel about doing these things? Confident, assertive, positive, good, clear, better than normal?

If the answer to your question is in the negative, then ask yourself why you feel this way and if it's appropriate, choose another thing to copy.Re-run the last three (or four) stages at least another 4 to 5 times to help you feel more comfortable.

Proactively choose a time when you can 'safely' practice doing this differently (i.e.: choose a situation that doesn't put yourself in the 'spotlight').

Ask someone you trust to give you honest feedback to assess your progress.

On the basis that any feedback is good; listen to the feedback (getting as specific as you can) and take action.

Take the last three stages again! As they say, practice makes perfect!

Agony Aunt

I'm the Introvert personality type so during meetings I like to listen to what is being said then go away and think about what was said. This does mean that I can be seen as not contributing to meetings, how can I learn to be more vocal during meetings?

Let's start on a positive note! Being able to listen carefully in meetings means that when you have something to say, its going to be relevant and valuable.

To help you feel more comfortable in sharing your thoughts with others, consider this stepped approach:

Ensure you read through the agenda and think about the possible things (topics) that might be discussed.

Next, clarify your thoughts on these topics (what's the situation, the perceived problems, what are the implications the possible solutions, who's involved etc. etc)

Make notes of your thoughts and refer to them during the meeting (which leaves you free to listen and choose your moment to contribute)

Take small steps. Commit to making at least one contribution to each meeting and the more often you contribute, the more comfortable you'll feel. It will take time, so be patient with yourself!

If it's possible, choose someone you trust to ask for feedback after the meeting.

Take whatever feedback they have (and the more specific and detailed the better!) and take action!

Repeat these steps until you find contributing easier; taking less conscious effort.

The Agony Aunt is here to help!

Got a burning question?
Our Secret Aunty is here to help.

Email questions to help@softwaretestingclub.com
We'll do our best to answer them in future issues.

We totally respect your privacy.
Questions can be kept anonymous.

Classic Blogosphere

Optimistic Developers, Pessimistic Testers

by Joe Strazzere

In my experience, developers tend to be optimistic folks, while testers tend to be more pessimistic.

- Developers are creators, with a natural optimism about making new things and solving difficult problems.
- Testers are fault finders, with a necessary skepticism and doubt.
- If developers are the yin, testers are the yang.

I believe this is a good thing, a sort of checks-and-balances tension that makes for better software.

But it does lead to some interesting contrasts...

Optimistic Developer: The glass is half full
Pessimistic Tester: The glass is twice as big as required

Optimistic Developer: This code hasn't yet been tested. It's not known if it has any bugs
Pessimistic Tester: This code hasn't yet been tested. It's not known if it actually works

Optimistic Developer: We are 90% done
Pessimistic Tester: We don't know when we'll be done, if ever

Optimistic Developer: We will refactor the code to make it better
Pessimistic Tester: They are throwing out the working code and replacing it with an unknown quantity

Optimistic Developer:
I only changed one line of code
Pessimistic Tester:
The entire system must be retested

Optimistic Developer: The code is the design
Pessimistic Tester: There is no design

Optimistic Developer: We'll fix those bugs later, when we have time
Pessimistic Tester: We never have enough time to fix the bugs

Optimistic Developer: This build is feature complete
Pessimistic Tester: The features exist; some are completely broken

Optimistic Developer: Anything is possible, given enough time
Pessimistic Tester: Everything has flaws, and given enough time I can prove it

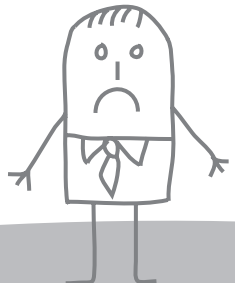
Optimistic Developer: Of course it will work
Pessimistic Tester: It might work, but probably won't

Optimistic Developer: One last bug fix, and we can ship tomorrow
Pessimistic Tester: Fixing this one bug will likely lead to two more

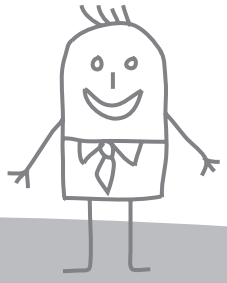
Optimistic Developer: Stop finding bugs, or we'll never be done
Pessimistic Tester: Stop creating bugs, so I can find them all

Optimistic Developer: There's no need for more tests
Pessimistic Tester: Let's just run a few more tests to be sure

Optimistic Developer: There is no I in TEAM
Pessimistic Tester: We can't spell BUGS without U



"It works on my machine"



Optimistic Developer: That's an "undocumented feature"
Pessimistic Tester: That's a bug

Optimistic Developer: I like to build things
Pessimistic Tester: I like to break things

Optimistic Developer: Sure, we can use the Beta version of this component in Production
Pessimistic Tester: We should wait until version 2.1

Optimistic Developer: Willing to bet that there are no more bugs
Pessimistic Tester: Willing to take that bet

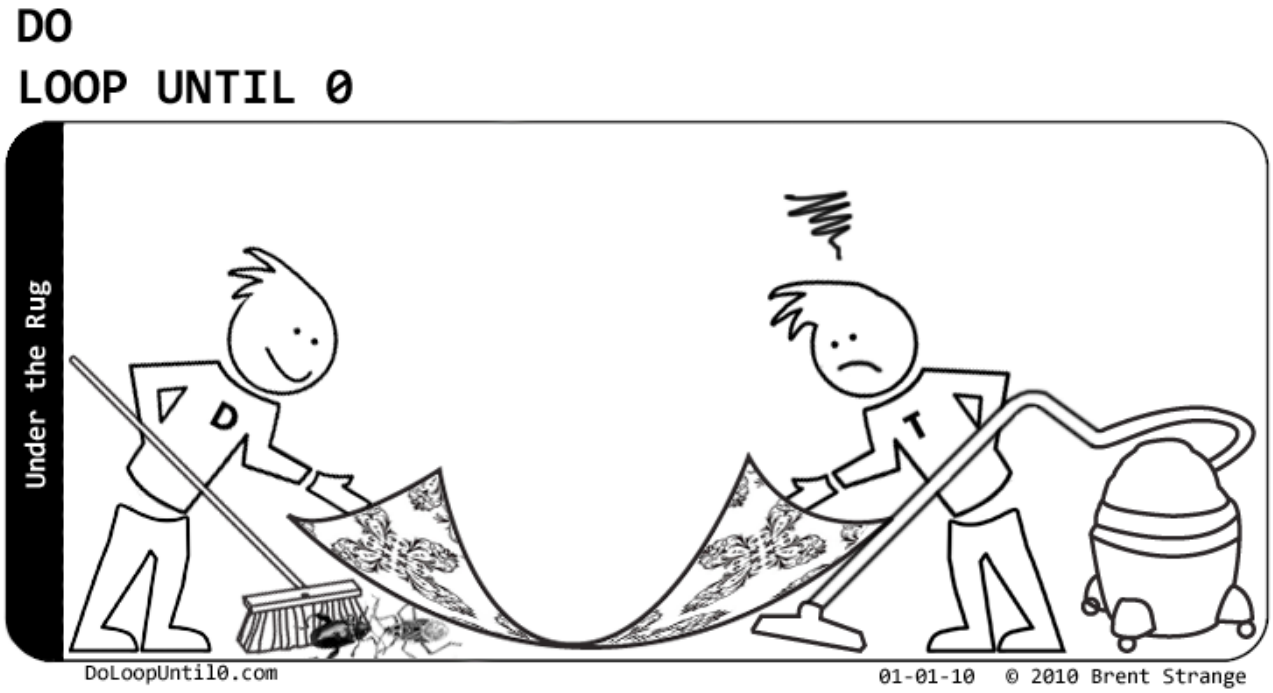
Optimistic Developer: Let's slip these changes in now, because I'm starting my vacation tomorrow
Pessimistic Tester: Let's not

Optimistic Developer: That will never happen in Production
Pessimistic Tester: Never is a long time

Optimistic Developer: It works on my machine
Pessimistic Tester: Perhaps your machine is the only one where it works?
Optimistic Developer: The sun'll come out, tomorrow...
Pessimistic Tester: Raindrops keep fallin' on my head...

Optimistic Developer: I'm a Realist
Pessimistic Tester: I'm a Realist

Source:
http://www.sqablogs.com/jstrazzere/1819/Optimistic+Developers,+Pessimistic+Testers.html



www.doloopuntilzero.com

BLOG OF THE QUARTER

The Added Value of Testers

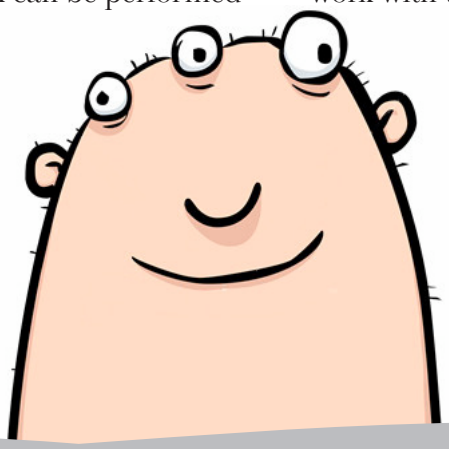
Ewald Roodenrijs

How can I as tester deliver an added value for my organization? Of course I check that the drawn up requirements and specifications are correctly processed in the software and that the software works correct. But more and more around me I see that this checking is rehabilitated. First there was the idea to automate the test execution software, then to outsource the work India when the automation didn't hold up to its expectations. Then the 'financial crisis' arrived. That meant that automation of test executions could show us again that it seemed to work and it did. Lately you hear more about Model-Based Testing, as I wrote in this post. Model-Based Testing takes over part of the test specification to be automated. What should I do? Where is my added value as a tester? Of course Model-Based Testing shifts the activities of a tester and the automation ensures that you can test more, which means that specific parts can still be tested by a tester (with more attention to this). But really, where is my added value?

That added value is what testing really is about. Testing is a combination of checking, exploring and accepting. So what are checking, exploring and accepting? See The W-model and Check, Explore and Accept for more details on this.

Is checking an added value?

When I look at the trends and innovations about testing around me, I see that checking is no added value for a tester. This work can be performed by inexperienced testers, be outsourced and even done by a computer. Because we only check whether the expected results match the actual results. In specifying on what the results should be when checking lies perhaps an added value, but an important part of the checking, functional checking, can de done with model-based testing (at least part of it). Naturally I as a tester have a direct involvement in creating the model. I can even add some test cases into the model to process any issues I think of. As a test designer/ architect I can even create the whole test model. Model-Based Testing only covers the functional part of a system. Other quality attributes should also be checked. A tester should create the test cases for these attributes. And also evaluations are an important part of checking. I think I have stated quite a few times how important evaluations are. When evaluating there is a clear added value of testers. Because testers look (very) critical at documentation. I as a tester can evaluate the documents in a good manner, but the effort of evaluations compared to creating test cases is however much smaller.



The added value of a tester in checking is:

- create models for model-based testing
- evaluate documents
- create non-functional test cases

Can a tester accept?

Can I as a tester accept? No! As a tester I'm usually not the future user of a product. So I cannot fully assess whether the solution is right for the user (in a UAT). This can only de done by the user. Accepting can also be seen as learning to work with the software and integrate it into business, but also getting acquainted with the System under Test (SuT). A testing coordinator can guide the accepting phase, but the real acceptance is done the users. The task of a tester is mainly a warning function; the user (client) self provides the acceptance. However by good cooperation (especially in agile proj-

Is exploring the added value for testers?

OK then, exploring. Exploring is something I as a tester can I do! I am an experienced tester and therefore have the test knowledge to do so. I can also ask the correct questions to get more information about the business and system. And also I don't take first response as the best, as a result I interview people about their knowledge. A checking tester will not look any further then the written down information. And I also have enough in depth knowledge about the business that I'm testing so that I know where to expect problems. As a tester I should therefore look for in depth information about the processes of my client and from there go testing. I can do this with test cases, but also through (a more creative mindset with) exploratory testing. Using this test approach I can show where the system needs to be adjusted, because the system doesn't resolve the problem of the customer. I use the specifications or requirements as a starting point and then build out the tests based on my knowledge of business processes and my testing experience. As a tester I can thus provide an added value to my client when I explore the software! To test better I need as much knowledge as possible of a client and its processes. Testers should be closer to the clients processes (business) rather than the technology. And of course specialists are needed that know more about certain technology (like security en usability experts). Besides our basic knowledge of testing, we need to have more business knowledge!

The added value of a tester in exploring is:

- create and executes test cases from other information than that is documented
- help the business with testing the business processes
- find those errors not derived from documentation
- have technology help in expert work.

Source: http://www.testingthefuture.net/2009/11/the-added-value-of-testers/

Funniest blog

You Want Good, Improvisational Agile training?

by QA Hates You

You Want Good, Improvisational Agile Training?

This ain't it:

Laughing. Overcoming embarrassment. Out-and-out goofy behavior. These are not normally the skills managers seek in their developers. But some agile development advocates believe these skills are critical for successful software projects.

A growing number of artistically inclined corporate trainers are promoting the principles of the Agile Manifesto with techniques from acting, improvisation and other art forms. Such exercises attempt to prepare software developers for changing requirements and other unexpected occurrences throughout the agile development process. Developers learn better ways to work together and how to put the team before the individual.

If you want to train them how it really works, you do the same thing I do to train new QA people: You cut the electricity, pull the fire alarm, and wait by the stairwell with an old American Gladiators padded quarterstaff you bought off of eBay.

Anyone who makes it out of the building can survive the ever-changing timelines and predictable, avoidable, but always sudden "emergencies" that, well, emerge. Anyone who knocks me off my feet or separates me from the quarterstaff gets a field promotion to Senior Test Engineer.

Or if you want to simply waste an afternoon, I guess you could hire these Agile training consultants for some entertainment and then go back to work the next day doing it the same way you've always done it.

Source:
http://qahatesyou.com/wordpress/2009/02/13/you-want-good-improvisational-agile-training/

Distributed Agile

The Key to Building Strong Distributed Teams

By Yvette Francino

Everyone is "going Agile" these days. Even though Agile Software Development methodologies have been around for awhile, there seems to be more and more buzz about them, offering a plethora of options such as the popular Extreme Programming (XP) and Scrum methodologies. New variations continue to pop up and many organizations seem to work in an environment that is kind of a hybrid of Waterfall and Agile, wanting to maintain the discipline of Waterfall and the flexibility of Agile. So what exactly is Agile?

A team of 17 technologists got together in February, 2001, and authored the Agile Manifesto:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
• Individuals and interactions over processes and tools
• Working software over comprehensive documentation
• Customer collaboration over contract negotiation
• Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

While the various methodologies outline detailed processes aimed at implementing these guidelines, the key to success lies in effective communication, collaboration and teamwork between business folks, developers and testers. Because communication, collaboration, and teamwork are all more easily accomplished from face-to-face time, typically it's recommended that agile teams are co-located – not just working at the same site, but actually sitting together as they work. (I've heard tales of XP teams being locked up in a conference room together until they finished a project, though the person that told me that tends to exaggerate. I'm sure the custodian let them out for biological needs.)

http://www.flickr.com/photos/timove/4207017343/

With all this emphasis on skills that are more effective in a co-located environment, is an agile team that is geographically dispersed destined for failure? With current trends of global teams, outsourced teams, and telecommuters, it's not always possible for teams to sit together. The prospect of flying these people to work together for weeks at a time is very difficult and expensive. With all the tools and technologies that we have available to us today, can't the same things be accomplished in a virtual way? It's important to look at why face-to-face teamwork seems to be more successful and to determine ways to foster those attributes in a virtual world.

Tools

Depending on your methodology, different techniques come into play that are more suited for face-to- face brainstorming. One technique, for example is a Wall Chart where various team members use sticky notes placed on a white-board or chart that is split into categories. Team members are able to easily move sticky notes between categories as changes are noted and discussed. Though some may argue you need to be face-to-face for this type of brainstorming event, Agile tools are available that allow for virtual brainstorming. A virtual Wall Chart complete with virtual stickies can be used. This is just one of many tools to facilitate remote collaboration. I would venture to say that today, with all the tools and technologies that are available to us, many of them free of charge, that we can easily find any tool to do the same kind of work that is done face-to-face. Using tools to document our agile processes has the added advantage of allowing us to easily archive our thoughts so they will be available for future discussions or teams.

Availability and Convenience of Immediate Communication

Another factor in the success of co-located teams is the ability to yell over to your next-door teammate "Hey, come look at this!"If you have to pick up the phone, or send an email, you are often a lot less likely to ask for help. Sometimes it just takes too much time to explain the problem-du-jour. It's much easier to show your teammate and work together on a solution.

Desktop sharing tools are now available for us to be able to share our working environment, so we can easily have a "come look at this" type of conversation with our remote team-member. Tools like Instant Messenger Chat allow us to see when our teammates are available and easily interrupt them, just like we do with our office neighbour.

Friendships and Trust

Perhaps the biggest key to a successful team is due to the friendships that develop amongst the team members. As we get to know one another, we learn who we can trust. We look forward to working together and helping one another, not just because of the company or the project, but

because we like our friends. We gain a feeling of camaraderie as we scratch our heads together, searching for solutions. We celebrate our wins together, going out for lunch or happy hour, patting each other on the back at our brilliant collective genius.

So, how can we emulate this on a virtual team? We need to learn to form friendships that are as strong as they are when working together side-by-side. This means we need to open up more and share a little of our "personal" self with our team-mates. We need to connect with them on more of a personal level. The social networking tools such as Twitter and FaceBook allow us to share more of a silly side of ourselves. OK, admittedly, some people take this a little too far. It isn't necessary to share our dancing-with-the-lampshade-on moments. But we can share photos and videos that are more personal in nature – family photos or photos of the hobbies or activities we do outside of work. When we do this, we get a sense of who a person is, beyond just their professional image. We find out things other than our work that we have in common, and we form friendships.

But uploading our photos and telling our team about ourselves is not enough. As leaders such as Dale Carnegie and Steven Covey taught us, we need to spend more time learning about other people than we spend talking about ourselves. Share of yourself, but engage with others on the team. Call them up. Joke with them. Tell them you appreciate their efforts on the team. Send them your "virtual smiles" in whatever way you can.

We live in a world that allows us to connect to people any time and any place. We have tools and technologies allowing us to communicate more effectively than ever before. Nothing can ever replace the value we get from a face-to-face smiles, but we can do quite a lot to foster strong communication, collaboration and teamwork. And however you communicate, whether it's via email, on the phone, or face-to-face, let your smile shine through. You are likely to find, not only a strong teammate, but another BFF (Best Friend Forever).

Do You Have Testing Cred?

By Anne-Marie Charrett

Every tester has a reputation of some sort. It may be that you're the quiet tester that sits in the corner and gets work done or the loud noisy one that's always got an opinion, but regardless of what you do or don't do you have a reputation. Rob Lambert wrote about some of the tester types and their reputation in his e-book Tester Types.

After completing the AST (Association of Software Testing) online BBST course on Bug Advocacy, I've come to the conclusion that the best type of reputation to have is TESTER CREDIBILITY.

To me tester credibility is the equivalent of having street cred. It's the X-Factor or Test-Factor. Other tester's look at you with a bit of awe and you command respect. You're the Fonz of the gang, the Yoda of the Jedi, and the Rizzo of the pink ladies.

Your bug reports are rarely rejected and people listen whenever you express an opinion or idea.

Cool hey?

So how do you go about getting this tester cred?

Write Great Bug Reports

Why the focus of bug reports? Because, as Cem Kaner describes "Bug reports are what people outside of the testing group will most notice and most remember of your work" .

Writing a bug report is not only about making them clear and reproducible, it's also about skilled analysis and communicating effectively with other project members.

The course challenges us testers to view the exercise of bug reporting a little differently. Instead of "bug reporting", the course refers to Bug Advocacy, or "Selling Bugs". Cem gives another great quote:

"If you want someone to fix your bug, you have to make them want to do it"

In the Bug Advocacy course, he also talks about:

- Presenting a bug in its strongest (honestly described) light.
- Presenting a bug in a way that connects with the concerns of stakeholders with influence—and if one particular stakeholder will be affected gets the message clearly.
- Communicating so well that your report enables good decision making.

Build influence

Very simply, every time you make a decision around a bug report you affect your tester credibility. Each decision brings in to call your tester judgment. Over time, the decisions that you make will start to have an impact on your credibility in either a good or bad way.

Again to quote Cem Kaner:

"Ask yourself what happens to your reputation if you:

- Report every bug, no matter how minor; to make sure no bug is ever missed?
- Report only the serious problems (the "good bugs")?
- Fully analyze each bug?
- Only lightly analyze bugs?
- Insist that every bug get fixed?

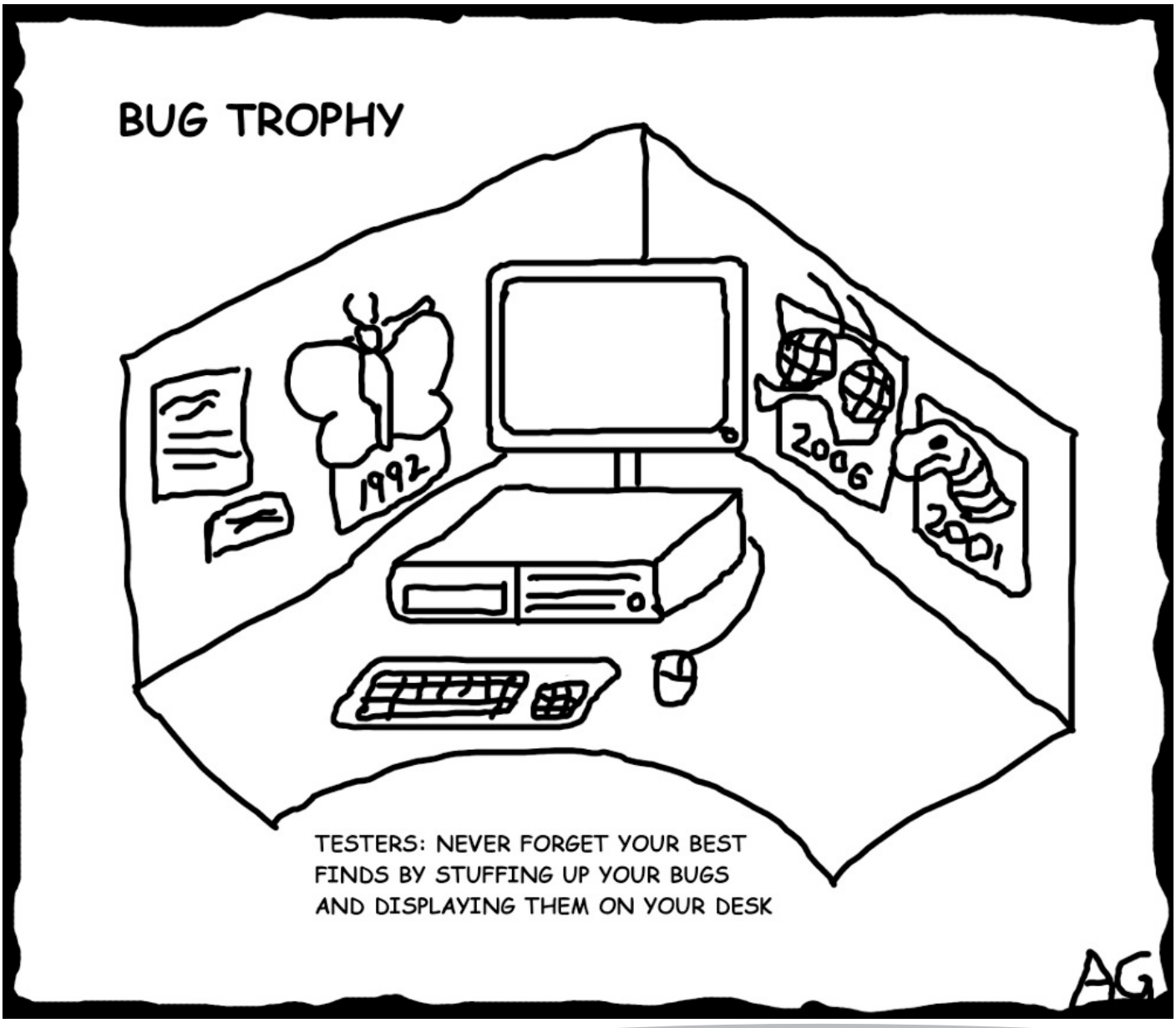
The comprehensibility of your reports and the extent and skill of your analysis will also have a substantial impact on your credibility."

A lovely example of a tester who has earned good tester credibility is by a Microsoft tester named Ricky Kurniawan. He describes in his post how he pursued a multi threaded issue that could only be reproduced on his machine. You can read the full story here A Bug Story: Multi-Threading.

By the way, there is a lot more in the BBST Bug Advocacy course, and if this has interest to you why not enroll yourself onto the four-week online course? You can find more information at The Association of Software Testing. The site's going through an upgrade at the moment, but you can still register your interest.

The information for this article was sourced from the Bug Advocacy 2008 course provided by the Association of Software Testing. Many thanks to all the instructors: Cem Kaner, Doug Hoffman, Adriano Comi, and Brett Leonard.

Anne-Marie Charrett is an independent software test consultant at Testing Times and blogs at Maverick Tester.



Twitter Conversation
By @skim and @qualityfrog

qualityfrog: @skim I got the strangest looks of disgust and annoyances once when I told a group that "doing scrum" did not make them Agile.

qualityfrog: @skim They were doing scrum poorly with CMM thinking. Defined tester role as quality cop. BDUF test plans. Repeatability was king.

skim: @QualityFrog absolutely. it would've been nice if we talked about this change as a group and talk about things that still need improvement

skim: @QualityFrog ...but *the person above* did leave his door open and i plan to take the opportunity to speak

qualityfrog: @skim They had separate dev and QA and management panels deciding how to implement Agile. Eventually learned to work together.

skim: @QualityFrog we just adopted the shorter release cycles and called it agile

skim: @QualityFrog anything "to deliver faster and more efficient"

qualityfrog: @skim A smaller hamster wheel does not make one agile. :D

skim: @QualityFrog we call them iterations, surely we're agile ;-)

qualityfrog: @skim Waterfall was intended to be iterative too. :)

skim: @QualityFrog but how does the water get back up?
qualityfrog: @skim You carry the part of the water that needs fixing or refining back up to the design phase. In leaky buckets? :)

skim: @QualityFrog that's just grand... ;-)

qualityfrog: @skim Or maybe the water gets hot enough that it evaporates forms clouds that rain back down on the top. :)

skim: @QualityFrog so we'll get rained on for a period of time...

qualityfrog: @skim It's about as enjoyable as living in a sauna. ;)

skim: @QualityFrog conclusion: working in a waterfall env is like living in a sauna with leaky buckets

Follow @skim and @QualityFrog on Twitter.

@testingclub

Conversation

"Starting out as a new tester"

with Rob Lambert, simon morley, Skim (aka Steve K)

Hello

Simon: Hi. Here's my first wave...

Starting out as a new tester - like the context-driven school says there are no best practices - the same is true about getting into testing or becoming a tester - there are no best ways how to do it...

Everyone is different, what works for one might not work for someone else.

Steve K: I agree. There is not only one way to get into testing. The approach will vary depending on the environment, people, etc.

Simon: I think on-the-job-training can sometimes be the new tester's first intro to pair-testing (actually sitting alongside someone and working thru a problem). Unfortunately for some it might also be their last experience...

For me, the biggest benefit for the new tester is getting their hands on a real problem - actually practising. They can make mistakes as well as find bugs in the app. Some bugs might be due to mishandling/configuration due to them being new - but this is all good learning sources.

Pair Testing

Steve K: I also think pair testing is a good introduction to testing with emphasis on exploratory/checklist style testing. I think it might be dangerous to jump too quickly into using automation tools right off the bat. First, learn what testing is about - allow the "new tester" tester to flex his/her mind, then later down the road, he/she can be exposed to automation checking.

Automation

Simon: Yep, avoid the automation voodoo to start with- If the company has some structured "training" even better - "testing" is such a wide open field that it can be overwhelming to start with.

Structured training isn't necessarily courses but maybe more of a checklist/guide of concepts and types of testing to experience - this probably helps both the new tester and the employer - have some sort of record of types that have been covered and what hasn't been covered. Some project iterations might need more of one type than another - it might be a GUI change, or input field modification, some network protocol introduction, etc, etc

So the checklist might start on a high level Static Analysis/testing, BT/UT/CT (basic/unit/component test), FT (Functional Testing) & ST (System Test - including non-functional testing) and also include domain-specific areas that the development unit needs - maybe some tool-specific knowledge, protocol-specific (eg HTTP, IP, XCAP, etc, etc) - All of these can be broken down and expanded depending on the unit - boundary-value, combinatorial, use-case validation etc, etc

This is a very incomplete list but already it's getting complicated - so it makes sense for both the new employee and employer to have a way of keeping track of some of the basics - one of the first review times it could be a task of the new tester to suggest what's missing from his checklist (after 2-3 months?) – so that the guide is personalised..

Checklist/exploratory: Yes agree completely. A very good approach (this works for not-new testers also) is to take a scripted test (not necessarily automated) and ask the tester if this test is still relevant, does it miss anything, where/how could it be improved - and if it is sufficient, why is it sufficient?

Here the mentor needs to be helping/pushing the new tester to think about what's going on and why...

I can't think of an environment where not jumping in with some OJT is not a good thing - obviously I'm not including any life-critical systems here - although it could still work there. The new tester will just have their work checked by their supervisor/mentor/helper more...

OJT?

Steve K: What is OJT?

Simon: OJT: On-the-Job-Training

Simon: So, what are the prerequisites, what's needed for a new tester to get going? Curiosity, willingness to learn, not being afraid of making mistakes??? Are these good starting points before later making more "academic" demands about critical thinking/reasoning?

Steve K: I think by default everyone has a level of curiosity and eagerness to break software. It's quite interesting to see how many people want to sign up for alpha/beta software to get the chance to try out new products/services and perhaps a bonus to be able to brag about it. I think "new tester" testers should be passionate, willing to learn, and definitely not be afraid of making mistakes (as you've mentioned). While some people think failing is a bad thing, it's part of the growing pains to become a better tester. We're human after all.

I think those are good starting points.

Simon: Yes, faults and mistakes (whether in the app or made by the tester) are essential for learning. I know I wouldn't have learnt so much about the systems I have worked on at the start if they were fault-free.

Interesting about alpha/beta testers - like you say some want to be first in line or (early adopters as I think marketing peeps classify them) but they don't always want to be finding the problems. Aren't some more "fair-weather alpha/beta testers" (only want it if it works - it's a kudos thing)? But, I think in general when you're employed to be testing then you have to want it, otherwise it's not going to work.

Simon: Think about courses and learning new topics - some people want to do a course/tutorial before practising, some want to get a feel of a new topic "hands-on" before taking a course and some don't want the course but want some "on-the-job" training - then there are others that just want to try it for themselves, let it sink in and see what they make of it - "so called exploring..."

The thing is that for some topics/areas one type will work for someone and then for a new topic maybe a different method is better for that same person...

Take "Google wave" - I've no idea if I'm doing this in a good way or not (I'm gonna stop/pause soon for fear of this turning into a monologue ;-)) - but let's just dive in and see what happens. I'll pick up some tips by observing how others do it - and see if they can be applied to me (or used by me). "Waving" seems to be a low cost or easy "entry-level" app so there's no need to read a manual or whatever - especially since the look'n'feel is similar to parts of other apps and there are parts that are intuitive - but that doesn't mean I'm doing it "right" or that I'm making the most of it....

Now back to testing - there's a bit of this that's an analogy to learning about testing...

Simon: So for a new tester, what are good resources online? Are some starting points better than others? Can some places be more intimidating than others?

Simon: On-line resources / learning aid ? Are there some that are better than others?

What about twitter? This seems to have a life of its own. I wouldn't recommend it for a new tester (or any tester really) as a resource for learning about articles.

Twitter

Rob: For me Twitter has opened up a whole new hive of contacts and feeds where I get loads of new articles/ideas and concepts. I'm not sure I would have come across these had it not been for Twitter.

Simon: True, twitter is a great resource - yes, there's things I wouldn't have come across otherwise. But I was thinking from a new tester's perspective - it could be overwhelming / information overload.
But, as with most things I'd recommend a "suck it and see" approach - find out yourself what you're comfortable with.

Rob: Yeah. New testers could become swamped by the mass of noise. Takes a while to filter it out before it starts to make sense.

Steve K: I agree with the both of you. Twitter has opened up an opportunity to listen and talk to other passionate testers/developers and insurmountable amount of information in articles, blogs, presentations, etc. The signal-to-noise ratio is definitely overwhelmingly for a new tester. Once the new tester familiarizes test concepts, he/she can start diving into the testing world in Twitter.

But where it is powerful (but still a bit random) is that you can contact someone who wrote an article or is knowledgeable in some area - if you're lucky and online they'll usually respond - so there is a sense of availability there. This works in other forums also - generally testers of all levels are approachable.

Rob: This is such a good benefit of Twitter. The ability to interact and chat (albeit it in small sentences) with people you would never normally have been in touch with is fab]

Blogs

Simon: Blogs? It's hard to recommend this as a way of learning about testing - many times they can be "parts of conversations/topics"

Rob: Agreed. Blogs are often very personal, often incredibly misleading and sometimes very political. Saying that though, they are invaluable sources of answers to questions

Simon: Q&A and discussion forums are a good way in - generally questions can be fired off and usually get a friendly response - usually many responses with lots of different aspects...

Luck Factor

Simon: Something that just occurred to me - there's a "luck" factor for the new tester also: luck that they begin in a "good" work environment that supports and encourages them. Can be intimidating to come in and think that you should be "questioning" and trying to make sense of everything that the unit/org is producing/developing - especially if there's some developers that aren't too happy with questioning from a new tester... Thoughts?]

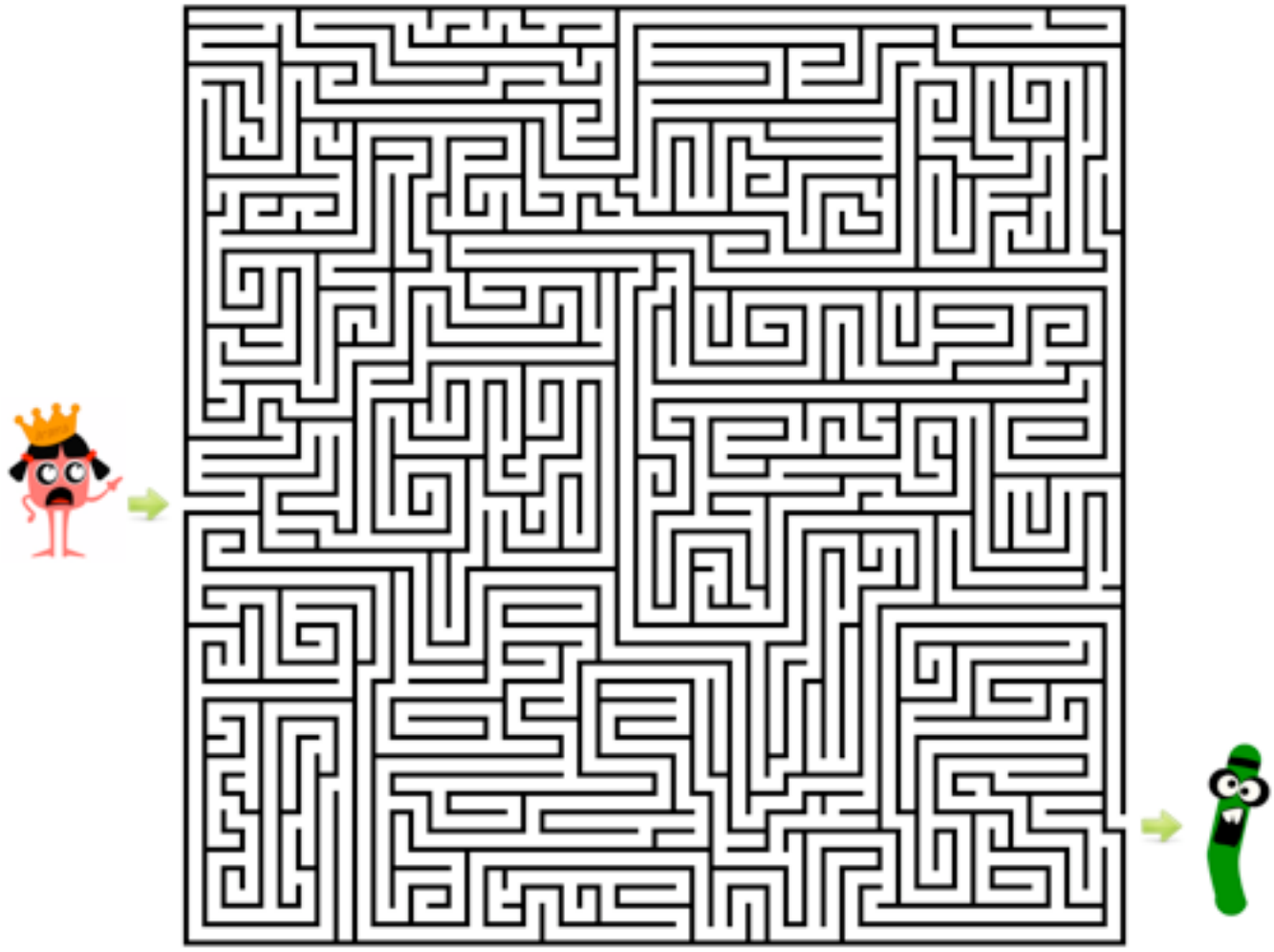
Rob: Couldn't agree more. In fact some testers can get stuck in a "bad" environment and not know that "better" ones exist. It pays to move roles every so often. Keeps your skills sharp and builds on your experience. After all, no two companies, methodologies or products are the same.

Do you want to take part in a discussion to appear here in the magazine?

Email rob@softwaretestingclub.

A Testing Maze

The Drama Queen has kicked up a fuss.
Again.
"Look, look, it's green" she mocks.
But is the joke on her?
She cannot reproduce the bug!
Can you help her find the green bug(ger)?!



Best solution uploaded to Flickr and tagged with "softwaretestingclub" wins a copy of Beautiful Testing*.

(* or equivalent in Amazon Vouchers)

Testing In The Open

By Ara Pulido

How to create a successful testing team for your free and open source projects

Selenium, Abott, Watir, ... Whenever you go to a testing conference you realize that almost any software tester uses or has used at least on piece of open source software to automate or organize their testing activities. But, what about testing open source software? What challenges does it introduces? What makes it different?

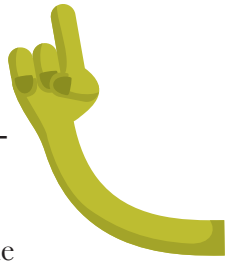
In traditional software development, testing is normally seen as an activity that it is performed behind the scenes. After the requirements definition, the design and the development, testing is usually performed by an internal group of people (or by an external one, under a explicit NDA agreement) that test the software and return results to the development team. Rinse and repeat. Even in agile environments, where the line between development and testing is fuzzier, this idea of internal knowledge about bugs or performance of the developed application still applies.

Free and Open Source Software (FOSS), however, is developed publicly. At any point of the development process, people can have a look to the documentation, get the code, compile it and try to make it run on their machines. Who would do that? Who would risk their machines using software that is not released yet and requires a lot of effort to make it run? A lot of people. Really. A lot. FOSS community is huge (and growing) and every user is a potential tester of your application. But, how do you drive this community to pursue your goals and make them part of your project?

Organizing the testing

Not every person has the skills needed to test software, but within the FOSS community there is great potential. One of the main problems faced by people willing to help out when they join your project is a sense of not knowing where to start or how to be productive and helpful. If you want to get back from the community you have to provide them means to make the most of their time.

Documentation



Development documentation can be useful for testing, but it is not enough. If you want the community to test your project you have to have specific testing documentation.

Some of the things that you should include is:

- A news page, with the latest information for testers. This may include a call for feature testing, the latest change-log of your project or a new hardware support that you have enabled and need some feedback from people owning it.
- A concrete and well explained way to obtain the software to be tested. You have to be specific, if you provide information about the source code, a tar.gz. file or a binary, people may wonder which one they should be using. Again, be specific.
- A publicly available bug tracker. People want to see how their bugs are evolving. If you ask your testers to email you the problems they find and you track your bugs privately, they will be discouraged, as they won't have information about their findings (have they mile stoned my problem? is it going to be fixed? was it a duplicate?). Having the bug you found tracked and scheduled to be fixed for the next version, can be the most reward-

- A very good "How to report bugs" document. Reporting bugs is a craft but there are tips or patterns that can be followed. There is a lot of documentation about good bug writing but write a customised one for your project.
- A web application to track results. Bugs are not the only product of testing, "Pass/Fail" reports are also needed. Allow your testers to access an application to track their findings.

Communication

In a closed development environment developers and testers communicate frequently through meetings, coffee breaks, after work beers, etc. If the developer cannot reproduce a bug, she can approach the reporter, who might sit in the next room, and he will reproduce it in front of her. We don't have that in a open source environment, so you have to communicate differently.

Provide an IRC channel to talk about your project. If your testing community is big enough, open a channel just for testing activities. Go there often. Answer questions. Don't make your testers feel lonely in their task.

Meetings are also a good way to keep up communication between developers and testers. Collaborators know when they are going to meet and there is an agenda (which should be open to add items to discuss), which helps driving the conversation.

"A nice way to encourage testing during an important time of the cycle are organized Testing Days."

The common place to have these regular meetings is the IRC channel of the project (of the specific channel for testing activities, if you have created one). If you have testers around the world, swapping between time zones to schedule the meeting is always a good idea. It would allow every member to attend at least one meeting.

Testing Days

A nice way to encourage testing during an important time of the cycle are organized Testing Days. Testing Days are scheduled testing activities that occur during a day (or a weekend or week) that has a specific and documented objective.

Let's imagine that you have added three new features to your application that you want to get tested before you release them. You need as many testers as possible involved looking for new bugs, regression issues or usability problems. Organizing a Testing Day can be the solution but you have to plan the day well to make it a success for your project.

One of the first thing you will need to do is to create a web page (or a wiki page in the project's wiki) with the information about the Testing Day.

You have to add as much information as possible. Your testers should be focused on testing and only testing during the day to make the most of it.

Here's some ideas for what to include on the Testing Day website:

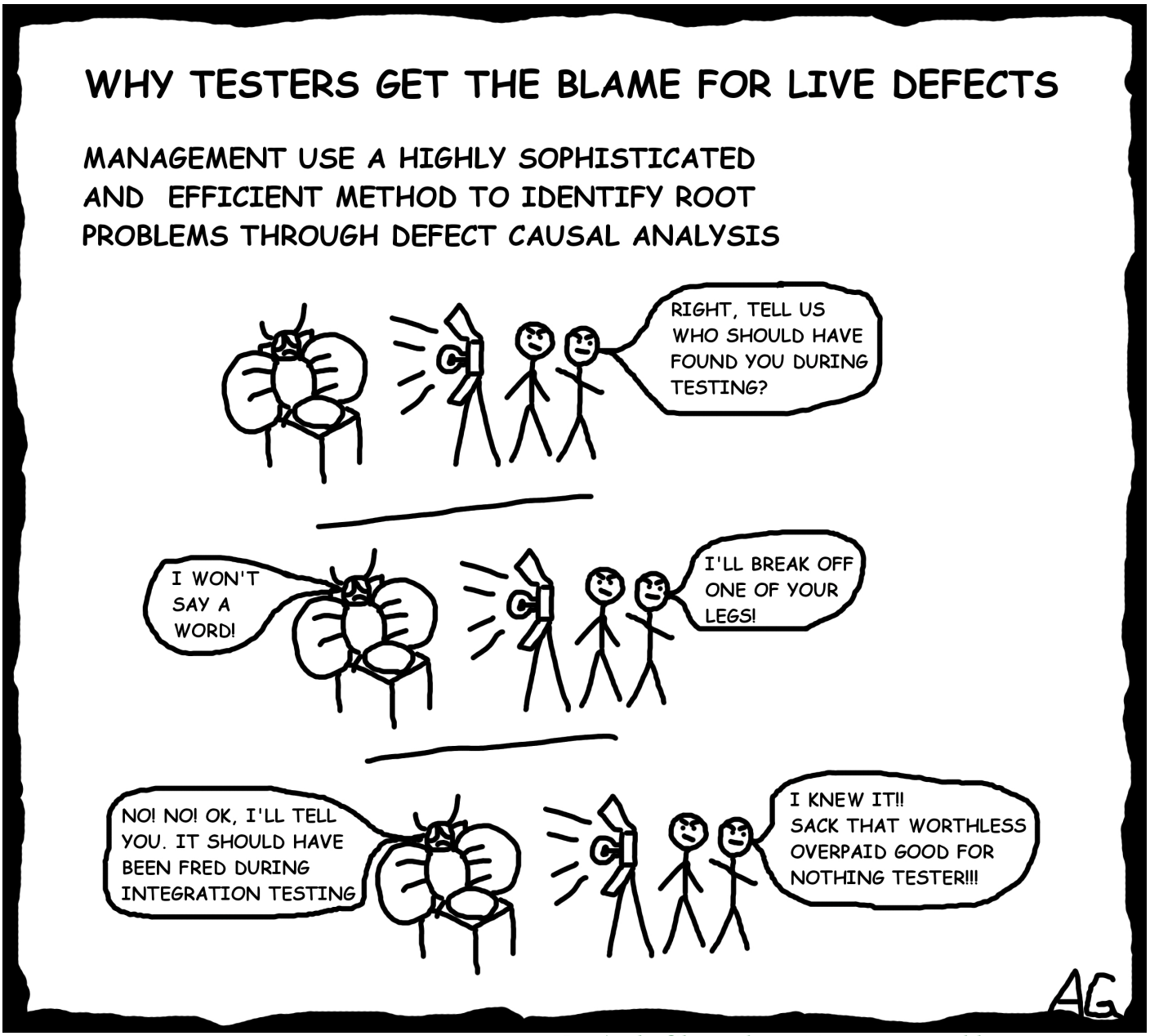
- Goal of the Testing Day. Are we testing any new features? Any specific hardware? Include any relevant information and point to documentation about the feature plus relevant installation files etc
- Include information about when this testing day is happening. Is it over a weekend? Only a day? Which time zone?
- What is the main medium of communication for that Testing Day? Sometimes Testing Days are the first contact testers have with your project, so try not to assume previous knowledge.
- Announce it properly to ensure a maximum success. Blog about it, tweet about it, announce it in testing and open source forums. More is always better.
- During the Testing Day ensure you are available (at least during your core working hours). Be at IRC, answer peoples questions, keep tweeting about it and promoting it.

Remember that Testing Days are a good way to gain new contributors. Make them feel welcome and productive. If they feel that their job is useful, they will come back. When you start accepting testing efforts for your project, you will probably see that some contributors are better than others; better testing, better organization skills, more commitment, etc. Start giving these people more responsibilities. Let them organize the next Testing Day. Accept their ideas and criticism. Make them responsible for the testing documentation.

If you present a nice testing infrastructure you will be able to obtain new testers from a pool of people already interested in technology and, sometimes, with a high technical and experience level too.

If you are a tester and want to start collaborating on an open source project, browse the project home page, talk to the developers and start creating this testing infrastructure yourself. You will be surprised how quickly the team embraces you as a critical part of their team.

Ara Pulido works for Canonical, Ubuntu's commercial support company.



By Andy Glover: http://cartoontester.blogspot.com/

Testers in the Gatekeeper role

By Markus Gaertner

In more traditional companies or projects testers and/or test managers are regularly associated with a gatekeeper function. James Bach, Cem Kaner and Brett Pettichord listed in their book "Lessons Learned in Software Testing" this lesson as number 12: Never be the gatekeeper. In fact there are a few problems with this view of testing.

Firstly it gives the impression of being responsible for the quality of the software. This is a double-edged knife. You will be blamed when the software still has bugs that you did not find and it will do, as exhaustive testing is most likely not possible. A famous english proverb states: "If I want a guarantee, I buy a toaster." (In case you found a way to be absolutely sure that your software is bug-free I'd be interested in a thought exchange.)

Second you will get to make decisions for the business stakeholders when fulfilling the gatekeeper role. As Jerry Weinberg points out in "Perfect Software … and other illusions about testing" testers provide information in order for the business to make conscious and informed decisions – not to make these decisions for them. Indeed this burden of responsibility will overwhelm most testers. This results in testers refusing to fulfill this role. If I got paid four times my salary, maybe I would make this decision, but would you pay that much to a tester or would you pay that amount more likely to the manager of that project?

Since just pointing out to "Lessons Learned in Software Testing" does not help while you're working on your current project, I decided to prepare a list of possibilities to deal with the problem.

Just do it

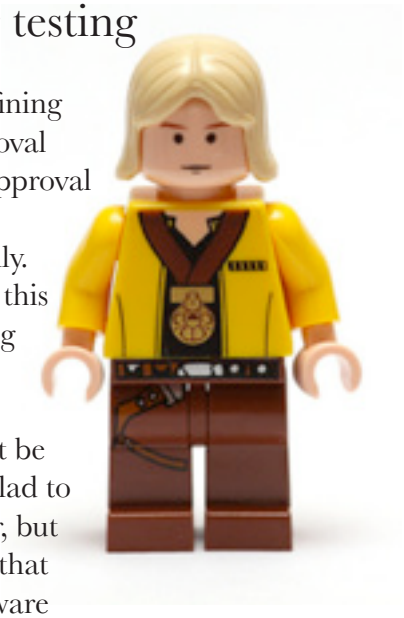
Sure, you can go ahead and just do what you're asked. But then you might of course run into the previously described problems of being the gatekeeper. On one hand you might take on responsibility that will overwhelm you. On the other hand you might get in the way of actually delivering any software at all since there will inevitably still be a bug in the software, that will do harm when feature X gets used. Maybe this is ok in your particular context, because your business made the decision to not use feature X for one year anyway. Thus you would be holding back the software for no commercial reason and your company does not make any money from it. Seriously, it's your name that is at stake if you stick to this, not mine.

www.flickr.com/photos/tim_norris/3717129497/

Define approval

More positively you can define what your understanding of an approval of the software is. By stating what you understand in your project would pass as approved, with the given information at hand, you define a basis for discussion. There will be people who view this differently. But by stating your understanding of approval, you transport the message "This is what I'm doing, there may be holes, but this is what I can ensure within the given timeframe." This is even better than an approval where the software is assumed to be perfect upon release.

Approve your testing



A special case of defining what you think approval means, is to define approval as having tested the software professionally. Cem Kaner pointed this out with the following comment to me:

"My approach might be a 4th option. I was glad to sign a piece of paper, but the signature meant that I had tested the software to a professional level. If necessary, I wrote that on the signature line. I was willing to say (or not say) that we had done a good job of testing, but you can have a well-tested bad product and a poorly-tested good product. I signed for the testing, not the product. "

In fact, one important step is to start thinking about the work you put into the testing. Most of the time you will not be able to tell whether the product is written well. Most testers simply don't understand the source code, the third-party libraries and the languages used well enough. Therefore signing for the quality of the software is ridiculous.

Both, software development and software testing are problem-solving activities. The development of the software solves the problem of the customer or user by automating certain tasks by the computer. Software testing solves the problem for the customer, in that she does not know if the delivered software really solves the problem in first place. So, we gather information about the software, which is then shared with the customer. Therefore it's reasonable for a tester to sign for the amount of data gathered from the product, but not for the product itself.

Refuse to fulfill the role

You can ask your superior to free yourself of the role. You can state "I can't afford to make this decision. I will continue to provide the information from our tests, but surely I don't know if I make the right business decisions by labeling the software with a stamp as vaguely as 'approval'."

Sure, your boss might get mad with you. In the best case you will get better information in the future in order to be able to make this decision (you should ask for a pay rise by then). In the worst case you will find yourself on a new job in a year from now. But you will send out the message that you don't feel comfortable with making this decision today. You send out the message that you do not want to put your reputation at stake. You point out that you know what you're doing but the decision is not yours to make.

Conclusion

Whichever way you decide you have to really think over your current working habits before making this decision. Do you have all the information you need to make the approval decision? Maybe you already know everything. You have worked half a year with the customer on-site to get the business insights, product knowledge, discover your customers needs and you are able to serve your current project with this information.

I'd be pleased to be able to visit your team for a week and get to know how this works. Since I believe this is an exceptional circumstance, generally spoken you should either send out a clear definition of your understanding of approval, like signing for the amount of testing you did on the product, or refuse to do it at all.

Markus Gärtner is a software tester for Orga Systems in Paderborn, Germany. Personally committed to Agile methods, he believes in continuous improvement in software testing through skills. Markus also blogs at blog.shino.de and is a black-belt in the Miagi-Do school of software.

Photo credit: http://www.flickr.com/photos/pasukaru76/4005368889/

Experience Report: Introducing Exploratory Testing

By Simon Morley

So you've heard of Exploratory Testing (ET). You've heard the PR and thought it sounds interesting. Yeah, that sounds like something I'd like to try.

Maybe you've read some of the literature in blogs (lots of people talking about it), websites (for example the context-driven school?), books (James Whittaker came out with one in the autumn) or some other buzzword being discussed in conferences or on twitter.

Whichever way that you've come across ET how do you get started with it and introduce it into a team or organisation? Well, this is something I did recently - so think of this as a case-study or experience report.

Why introduce it now?

There was a need to test/assess a feature in a more complex environment (middle picture below) as there were problems doing it in the full customer-like environment (right-hand picture below.) Here was an opportunity to re-use some previous experience of ET and introduce a "structured active learning/test" activity with the emphasis on quick start-up.

Our goal was to assess the big picture (below, as well as smaller individual configurations) and conclude if we should do something, how we should do it and the when was ... "as soon as possible, if not sooner". The "if" we should do something was in the context of the other testing that had already been conducted on the feature.

Don't go overboard on the "process" or structure of the work. Start gently, learn and adapt the "process" to your own needs.

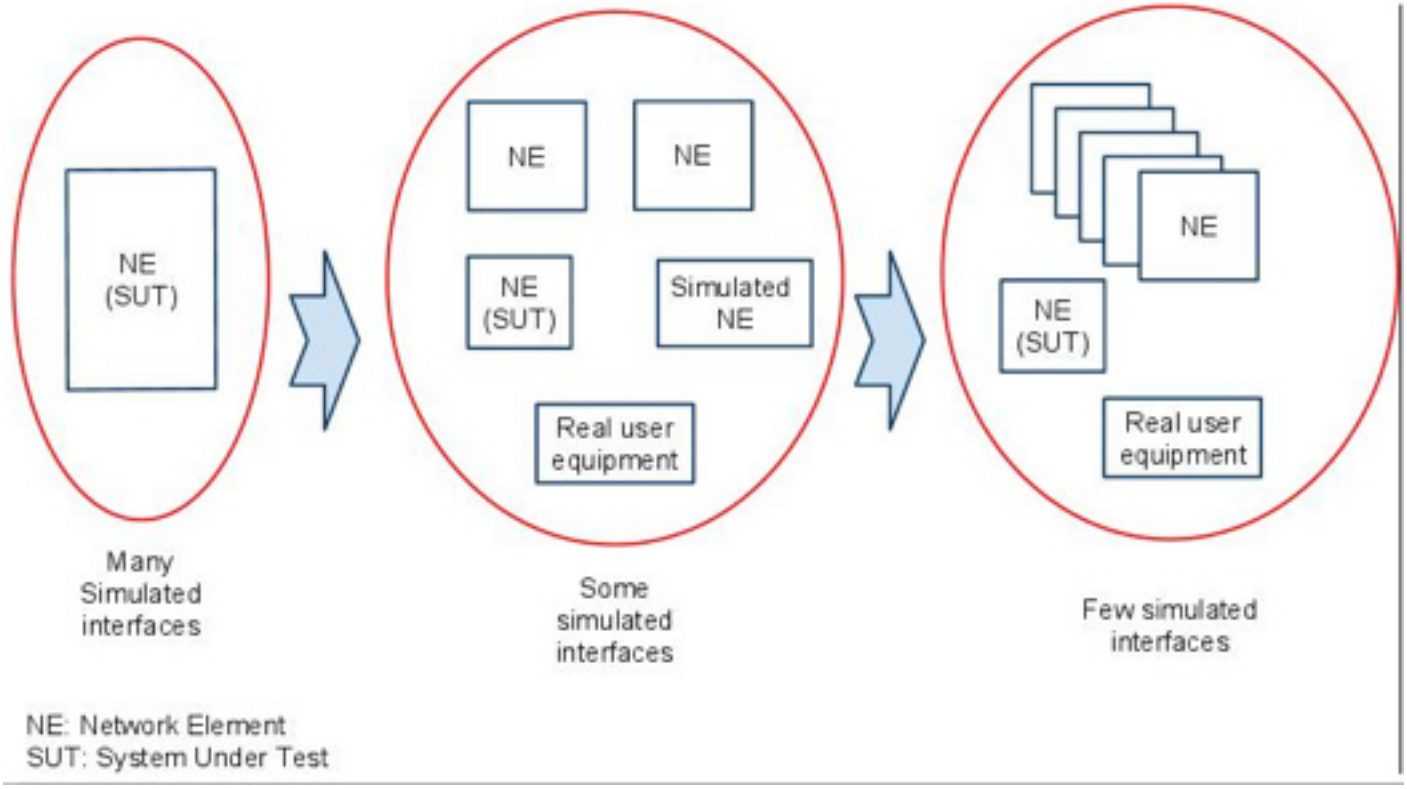
Time for a new intro? Analysis...

We set up a small team that started analysis on the target feature - what it meant in an isolated node and what it meant in a larger system. This was an opportunity for us to get to know the feature and other aspects of the environment (there were multiple features changing at the same time.) Besides the test team we involved various designers and system managers to understand different aspects of the "problem".

With regard to the picture below we were looking at the middle environment - a "small" network where a single network element (NE) was implementing a range of features and we were looking at interactions between NE's as well as the core use-cases in the NE system under test (SUT). The main reason for introducing the activity was that the target feature interactions were due to be tested in the "fewer simulated interfaces" environment.
After some sessions involving different types of questions, investigations and follow-up sessions we came to the conclusion that there were some "interesting" cases/scenarios to look at. These were areas of the functionality and system configuration where we saw that we could add some value - we identified some "risk areas" - areas where we thought "well we don't know if this will work when we plug in this piece of hardware".

The whole analysis phase was looking at different aspects of the problem - understanding the end-user requirements, simplifications we were making in the environment (e.g. aspects to consider by simulating parts of the network) and what it might cost to test in the environment we wanted.

The analysis ended with a "go decision" - we had an OK cost for the simulation effort and we'd justified the need for the testing by identifying some "risky" areas that we thought we give us some benefit to explore.



ET in action

When we started the activity I didn't announce "this is going to be an ET activity or we're going to do some ad-hoc testing". I framed the work in terms everyone was familiar with: We started with an analysis activity, evaluating some useful/valuable use cases and using those as a starting point to investigate/test in an existing lab.

My task was to keep the simulation development activity in synch with the test execution and other lab scheduling, discussing approaches to testing different scenarios and ideas for further investigation.

Another aspect of my role was more of a "facilitator" - guiding the group, maybe setting some of the starting points and being there as a ball-plank for ideas, but it was the team that were making the interesting discoveries about the product - the unexpected feature interactions which turned out to be bugs! I was their to trigger some of the initial curiosity and enthusiasm about exploring the product but it was ultimately the team that did the exploring.

Time to stop

So, how did we decide when to stop? We had sketched some key use cases that we wanted to test (know more about). After we started the first test sessions we were able to re-prioritise those scenarios, change some, add some more scenarios and begin to add some test design details. But it was this prioritised list that was our main driver - we also had a certain time window in which to operate.

Lessons?

Frame the problem. Get to know the problem. This is a great opportunity to brainstorm (one of my favoured starting points).

Involve a wide scope of people to capture lots of different aspects - understand the problem/feature from the design perspective as well as the end-user/stakeholder perspective.

Try to understand/estimate the costs of the environment/configuration you want to work in: what simplifications/simulations are being made - does this make some test scenario unrealistic or not worthwhile, is the cost of the simulation going to delay the project..

Process/Structure

Don't go overboard on the "process" or structure of the work. Start gently, learn and adapt the "process" to your own needs - this may be more about adapting your existing

work processes so that they are more ET-like. Take as big or little steps as you feel comfortable with.

Set out your own objectives and following-up "how" you meet them (you may decide that something is not working and it's time to re-think how to look at the "problem".)

Be sensitive with the team dynamics. Relate what you're doing to new ground - this has worked for me in the past.

In the beginning I didn't have all the details about how I wanted to run the ET trial. That's OK - it's more about being able to adapt as you go. I started the team from the analysis viewpoint. We decided there was a "business case" to look at some scenarios - and the team focus became how do we learn about these use-cases and test them rather than "how to we follow our existing/known processes to do this work".

Change is good?

Not everyone was comfortable with a "less-structured" approach. But it wasn't really less structured - it was just structured in a different form. We had regular follow-up meetings, we documented our progress and issues in a common/centralized way - we had people available/working different days - so it was important to have a central DB of issues, questions, results and we supplemented this with email updates to each other (keeping everyone in the loop).

It's infectious!

Once you start a team looking a problem from a new angle it doesn't take long for the ideas to be self-generating. Our testers have a wide range of freedom in following up "hunches" but this really took off in this exercise. The team would change tack during the test session quite dynamically. We evolved a set of target tests that we wanted to perform but these were added to and modified quite dynamically.

"Not everyone was comfortable with a less-structured approach."

Learning is good!

We learnt about areas of the product/function that hadn't been seen in some previous test phases - partly due to using different environments, tools and configurations (these weren't available to the other phases.) We found some key interaction bugs that would not have been so pleasant to a customer.

There is also a key learning here - you have to want to learn and explore! If you want to try ET just for the sake of it then it's probably not going to work.

Summary of things to think about:

High-level use-cases are the starting point

Understand your scope/environment/configuration and how that relates to the main use-cases

Don't put more effort in the up-front test design than is needed to get started

Experience in ad-hoc/ET is good, but getting started and learning from mistakes is just as good

Document what you do - the whole activity is a feedback loop - you may discover some use-cases are no longer so important for the scope you're working in

As Nike say, "just do it".

Would I recommend and do it again?

Absolutely.

I think it's good for organisations to try out new ideas and for them to put their own spin on it - to make the activities "theirs". The activity is intellectually challenging - you need to have some good objectives before starting and even be able to motivate the need for it. It is also challenging during execution to switch course during, follow-up on new leads or try a new scenario or "what-if" case.

So, yes, give it a go...

This article was by Simon Morely. He has a blog: http://testers-headache.blogspot.com/

Tester's Diary
by Ainars Galvan

October, 5th
Hello,

I don't get it! Am I really awake or is this a nightmare?

I can't believe they fired the whole team before we could even open our mouths. How could he have meant it?

"Testers don't provide any value for the company, and you only delay the whole release process…"

Cool down, Jane!

It's not like this is the first time you are unemployed. Remember the 9 months after graduation looking for a programming job…

Like many of my friends I went to study Computer Sciences at University, and to tell the truth I very much liked what I learned. Those were also some of the best years of my life with all the nice experiences and the good friends I made.

After 9 months "outside" I understood that in order to find a position as a programmer I needed something they didn't teach in school: WORK EXPERIENCE. Talking about this with some friends they told me to start as a tester and move up from there, and I thought to myself "why not?"

To cut a long story short, I went to Amazon and bought a book about testing. The "knowledge" from this book and my degree helped me get a testing job at a mid-sized development firm; and here I am as a 6 month QA veteran… unemployed again… and still no experience in programming.

So now I'm looking for a job, the sad thing is that talking to some colleagues it appears the same problem happened to others too.

I hope this diary will help me with my career decisions and development.

October 9th
Hello diary, I have good news. After applying for a couple of testing jobs I got hired in "Screws, Bolts & Co ". I start next Monday.

"I still don't get why they started calling me The Enemy."

October 12th
My first task is to test a juicer. It's funny but when I asked about it they told me not to write any test cases and just look for the bugs.

I disassembled the juicer in a search for bugs but did not find any… It was quite late when I tried to assemble it back together and failed. Since everyone had already gone home I wrote a report of no bugs found so far.

October 13th
What a mess!

My boss misunderstood my message and took the juicer to a demo. The smoke tests failed literally and they had to open all the windows to clear the demo room.

Lesson learned for the day: unit testing is not enough…

October 14th
I was officially notified that this is my last opportunity to make it right!
Now I'll be testing more thoroughly. I've got a few books on juices and last night I read them all from cover to cover.

My Boss was surprised to see an invoice for $200 for various fruits and vegetables, and after some arguing we settled on a budget of $150 for the project.

October 16th
The Engineering Team is angry at me since they can't find a way to solve the problem I reported with the coconut juice. For me this is weird since coconut juice is just a juice, but I must be missing something.

I still don't get why they started calling me "The Enemy".

October 20th
I am again looking for a job…

While engineering was trying to address the coconut problem, the competition came up

with the revolutionary "Citrus Juicer 2000", and all our potential buyers went out and got one.

On top of that, my boss found proof that coconut juice is actually called coconut water and it doesn't qualify as a juice.

Whatever…!

October 28th
The Work Placement Office sent me to a retraining course. I have now been certificate as an "Electronic Device Tester" or an EDT as they call it for short.

I also hold a certificate stating that I know how to use the Electro-Efficiency Measuring Device called "Electron Runner".

This certification really opened my eyes - I now understand how to work based on a 27-step process to assure any given electrical device is safe for the consumer.

November 2nd
Now that I am certified it was actually easy to get job.

However my new issue is that everyone assumes I know how to solve the problems without any assistance.

For my first assignment I need to solve a showstopper issue: Water leaking from one of our Steam Iron models, and I can't apply anything I learned in my course.

The leakage only takes place when the iron is so hot that the water evaporates before we can understand where it's leaking...

November 3rd
The Chief was so happy he invited me to lunch!

I couldn't sleep last night, thinking about the damn iron leakage. Watching TV there was an advertisement of baby diapers where they colored water to demonstrate how it absorbs liquids… EUREKA!

On my way to work I entered a bookstore, bought some blue ink and colored the water for my test. Iron's leak was found immediately and the error was fixed before noon.

November 6th
I was invited to speak at a local Conference, explaining my ingenious ideas.
Bought a whole box of ink at the expense of the company.

November 10th
Found a regression problem with the Iron's steam regulator. When working at low levels it stops producing steam at all.

The engineers are struggling, saying they can't reproduce it in their environment…

November 11th
Problem is still not reproducible, and it appears to have even worse effects during long runs – when working at low levels for more than half an hour the steam production mechanism breaks completely.

November 13th
I'm a dead woman!

It turns out the problem was caused by my idea with the ink blocking out the steam feeding tubes.

They say no human would ever pour ink into an iron and rejected my problem.

Now every one's angry at me for causing unnecessary stress and a delay in the project.

November 16th
I have been reassigned to testing pens…

They say I deserve the task. There is no automated manufacturing and each pen is unique so each of the 10 000 pens needs to be disassembled, re-assembled and then I need to draw a few lines.

At least I still have a job.

November 17th
My Boss came laughing and said that everything was a joke.

Their best practice is to test only 3 randomly selected pens from each batch of 1,000 and if one is broken the whole batch is scrapped.

It appears that the pens are made by computers or more precisely, robots.

The process is highly repeatable so if 3 pens are fine, there is little chance that other pens will be broken within a batch.

November 23rd
Today I was appointed as the main testers in the new project: a Fridge.

I have no idea how to test a refrigerator!

The Project Manager said this is fine since no one knows how to do it anyway.

Given the importance and size of the entire project I started a deep google-based research of the subject.

November 26th
I'm reading some sites and books on the Art of Cool Engineering.

Engineers began a discussion about options of automatic defrosting.

I wrote an email to my boss requesting we create a separate test room with advanced temperature control since studies show that room temperatures have high influence on fridges.

November 30th
Today the engineers argued whether doors must open to the right or to the left. Some suggested we create two models. Peter said that a "Universal Door" is also possible to construct. But the others say it is more expensive.

I got a room with a powerful air-conditioner. They ordered a heating system as well.

December 7th
It turns out that we are supposed to create fridges for the storage of ice cream in supermarkets and they have horizontal sliding doors.
I was checking the documentation but found no requirements for defrosting.

Peter came to borrow my trash bin because his is full of outdated design models.

But mine is also full of old testing checklists that make no sense any longer.

December 8th
I had all day today to rewrite my testing checklists and even add some optional checks in case I have more time at the end.

There is nothing to test so far.

December 10th
I volunteered to collect all the waste paper for recycling. With a delay to the delivery of the heating system I have had a new idea on how to reuse the waste paper for high temperature tests!

Anyway, at least I will keep myself busy and see what's going on.

December 11th
The paper collection volunteering was a good idea!

I saw what they are doing in engineering and I realized I misinterpreted quite a lot of the requirements.

Now I need to go and re-do my checklists once again.

December 16th
When the boss "discovered" what I've doing the last two days he got suspicious.

He made copies of all my "wrong checklists" and disappeared.

I'm almost done with the rework to my check lists.

December 17th
Go figure!

Apparently half of my "misinterpretations" were actually engineer misinterpretations.

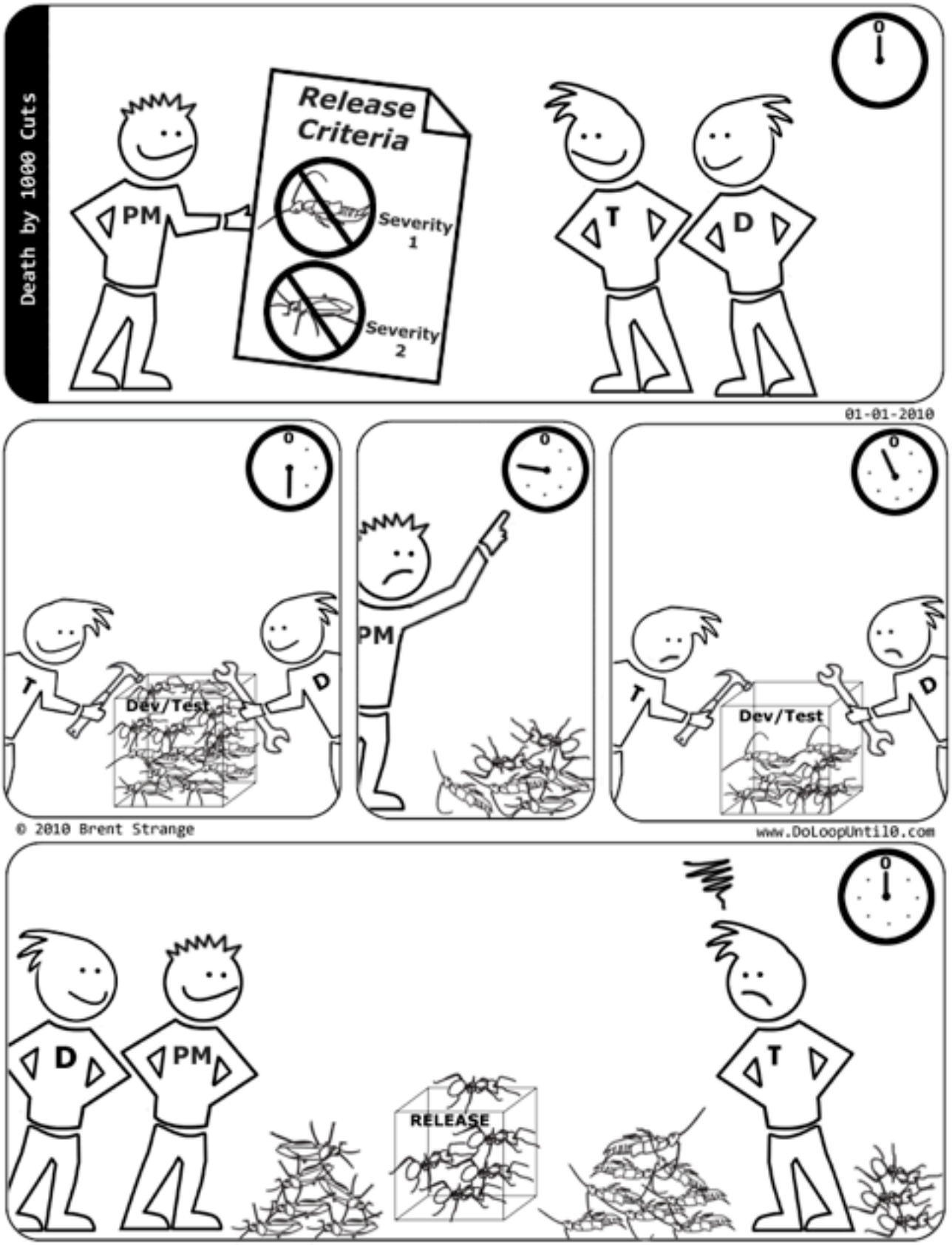
December 18th
During this mornings Stand-Up meeting Peter said that I ruined the project by asking the right questions.

Now they don't have time to make everything work right.

It was strange, but I was the only person to laugh at his joke.

You can find Ainars on the Software Testing Club.

DO
LOOP UNTIL 0



What is Social Media?

and How will it impact you in the future

by Ant Gardner

In recent months, perhaps the last year, we have been inundated with talk of a new wave or fad of 'social media' but what is it, how will it affect us and is it really new?

Well no, social media is generally accepted as the Internet tools for sharing and discussing information which therefore includes blogs and wikis which have been part of the world's culture for some years. The difference is that originally people started writing niche blogs that appealed to a niche audience; 'social media' has now broadened its horizons and started to penetrate the masses.

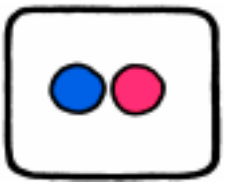
Friends Reunited was launched in July 2000 had 2.5m users by the end of 2001 and over 15m by 2005. This and similar sites in the US and Australia started a huge Internet trend towards connecting people socially, not just businesses but everyone.

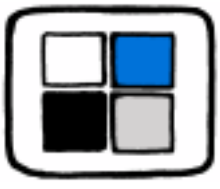
Since Friends Reunited there has been an ongoing evolution of social sites connecting people with perhaps Facebook (200m Users), LinkedIN (40m Users) and Twitter being the latest incarnations. So what? How does all this affect you? Well, it affects you because social media has fast become the communication media of choice! The masses are starting to centralise their lives around services that are cheaper than SMS and less cumbersome than email. You may not like it and you may not accept it but perhaps you should take a closer look!

The 'social media' phenomenon is here to stay, of that, there can be no doubt. It will surely evolve further over time just as it has from the inception of friends reunited but it certainly won't go away. Already, various platforms for social media are gaining traction within the online community. If you want an answer to a question then the chances are you'll get an answer from one or more of the many twitter users within minutes of posting it. If you want feedback from your business users then the chances are you'll get it quickly from something similar to Yammer and if you want to share your findings then the chances are a wiki will serve your purposes most of the time.

What does this mean to the testing community? Well, as a tester, you have a huge wealth of information available. You have a huge number of testing, development and methodology related groups available through social media to answer your questions or just network with other like minded people. You can even find new ways to engage with your end user through social media.

Over the coming months, the Software Testing Club magazine will be looking more into the various aspects of social media and the implications on software development practices.





Small Ads

Atomic Object. We build ridiculously good custom software for startups, huge companies, and everyone in-between. Our testing, process, and user experience fanaticism will win you over. Visit us at www.atomicobject.com or give us a call at (616) 776-6020.

§

Selenium RC users: you owe it to yourself to try Sauce OnDemand (cloud-based Selenium RC). Accessible via standard Selenium RC API. 10 popular browsers. As many parallel threads as you can handle. Special STC offer: 50 hours free testing: register at saucelabs.com using work email; append "-STC" to your account name.

§

Testuff. Manage your testing. A full test management suite, easy to use, intuitive and fully featured. Integrates with all bug trackers and automation tools. Unique video recorder, test runner and much more. Try it now for free - www.testuff.com

§

Software tester looking for opportunities where testing can be done from home. Thorough & efficient bug detection. Have experience with uTest.com. Please contact by email: vlights@e-graphx.com

§

We test, analyze, monitor and improve Web performance. Customers with the highest possible performance goals use Apica LoadTest™ and Apica WebPerformance™ to test their applications' max capacity and monitor their daily performance. The products in the Apica WebExcellence™ Suite are provided as Software as a Service. Contact mike.howse@apicasystem.com

§

Improve quality through manageable traceability and visibility in test execution. For multi-platform, automated and manual testing. "Nuevosoft Test Manager" http://www.nuevosoft.com

§

Curious about the next steps of your daily software testing activities? Are you interested in what the future brings you?
Go to http://testingthefuture.net and see what will happen today and tomorrow with software testing.

§

FREE ISEB/ISTQB Certification in Sofware Testing in London
This is a 6 day Course which includes "Maximising your employment potential in Sofwarre Testing" and the Certification
Contact the Working Men's College for details:
020 7255 4722 or r2r@wmcollege.ac.uk

§

FREE Web Automation Library and Macro Recorder for Internet Explorer. Automate testing of complex Web/AJAX/DHTML applications using the language of your choice: JScript, VBS, C#, VB.Net, C++
Download Twebst Automation Studio: www.codecentrix.com

§

XStudio (http://www.xqual.com) is a FREE test management application handling the complete life-cycle of your projects: users, requirements, specifications, scrum projects, SUTs, tests, tesplans, test reports, campaigns and defects.
An Open-source (LGPL) SDK allows to develop your own launcher to interact with your own proprietary (or not) tests.

§

Learn how to save time when testing web applications by creating robust, reusable automated tests using the opensource tool Selenium. Our 1-day workshops are practical and "hands-on" so bring your laptop.
Delivered by one of The Testing Geeks, Anand Ramdeo.
To book email debra@electromind.com or call Steve Allott on 07734 761363

§

TOSCA Testsuite™ enables testers to define and maintain automated test cases in a business conform way – without scripting or programming. To learn more, visit www.tosca-testsuite.com

§

It works on my machine.

§

Small Ads

LogDigger simplifies and accelerates the process of reporting bugs in a Web application, freeing up the testers' time, while providing all the details that developers want. It allows testers to report an issue in 30 seconds, complete with usage history, server logs, and an annotated screenshot. http://logdigger.com

§

QMetry is a SaaS based comprehensive test management platform that provides a tester friendly interface, and provides powerful capabilities that help QA teams to integrate, collaborate and co-ordinate the entire testing process to increase testing effectiveness and efficiencies. Contact: support@qmetry.com

§

We can help you test your applications and upgrades...web-based or in-house! Tools, techniques and results at a reasonable price. 10+ years of testing background. Lots of contacts and projects. Let us help. Collin Klepfer 970-217-3423

§

Net-Translators is one of the leading translation companies specializing in software localization, including GUI, Online Help, documentation and website in over 60 languages. We supply services to large multinational software companies. Net-Translators is certified ISO 9001:2000. We maintain branches in the US, the UK and Israel
www.net-translators.com
sales@net-translators.com

§

Do you need Software Testing Services but have a Tight budget? Visit us at www.100DollarsTesting.com. Our unique model brings down the testing cost which no other Company can match, while there is no compromise on quality of testing as your application gets tested by ISTQB certified testing professionals.

§

Are you looking for a test automation solution that provides definitive HIGHER ROI?
If you need a well thought of test automation strategy, which can ensure that your software product achieves: Better Quality, Faster Time to Market, Lower Testing Costs.
Visit http://www.impetus.com/qlabs/testing-automation-services.html or email at isales@impetus.com

§

Beyond Certification is a network for professional development aimed at gaining skills in a Distributed Agile environment. Useful for mentors, job-seekers, employers, or anyone wanting to learn and practice Agile skills from a distance. http://beyond-certification.ning.com/

§

KiwiQA Services - World Leader Software Testing Provider
Manual,Automation,Load,Performance, Security Testing Expert
Visit www.kiwiqa.com for more info
E-mail : niranjan.limbachiya@kiwiqa.com
Free Testing of Any application for 10 Hours !! So Send your Inquiries.

§

FREE WHITEPAPER: Seven deadly sins of test automation. Download now at: http://www.origsoft.com/whitepapers/seven-deadly-sins

§

Exploratory testers and accessibility testers required with minimum 3 years experience for contract and permanent positions in Central London. Must be able to work on own initiative. Strong HTML and CSS knowledge essential. Contact steve.green@testpartners.co.uk

§

QA Automation is becoming the order of Test framework. But the challenge is to take an informed decision about the Right Tool. eggPlant from TestPlant.com is a GUI Automation Black Box tool that test like HUMAN EYES. Please visit www.testplant.com for AWARENESS.

WANT A SMALL AD?

Future editions will include small ads from paid members only.

Visit us online to find out more. www.softwaretestingclub.com

We like ideas.
Submit yours.
www.softwaretestingclub.com