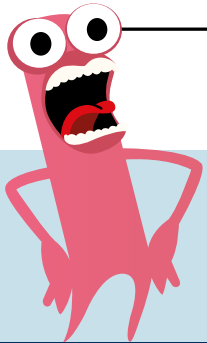




THE TESTING PLANET



October 2011 | www.thetestingplanet.com | No: 6

£5

Finding the best tester for the job

Testing the tester - a short story about practical recruiting methods for testers - see page 22



Changes that are out of this world - what does the future hold for the profession of testing?

The future of software testing Part one - Testing in production

By Seth Eliot

Hey, want to know the future? OK, here it is:

- Testing in Production with real users in real data Centres will be a necessity for any high performing large scale software service.
- Testers will leverage the Cloud to achieve unprecedented effective-

ness and productivity.

- Software development organisations will dramatically change the way they test software and how they organise to assure software quality. This will result in dramatic changes to the testing profession.

Is this really the future? Well, maybe. Any attempt to predict the future will

almost certainly be wrong. What we can do is look at trends in current changes - whether nascent or well on their way to being established practice - and make some educated guesses.

Here we will cover Testing in Production. The other predictions will be explored in subsequent editions of Testing Planet.

Continued on page 2

The mobile environment

By Karen N. Johnson

When we're testing a mobile application, most of our focus is centered directly on testing the application. But there are other considerations surrounding our applications to consider such as the installation of the application, on-going phone and application upgrades, as well as how our application interacts with other variables in the mobile environment. There are still more variables including firmware upgrades, memory card space issues and device permissions and settings. If you've been in software testing for a long time, these ideas (install, upgrade and interaction testing) seem familiar, even a bit like déjà vu. How would something new such as mobile seem old and familiar?

Back in the days of client server application testing, we had to think about installation testing. In the past decade, we've been primarily focused on web testing where installation testing is rarely needed. But in the 1990's we were focused on desktop software and how gracefully applications could live and perform in a PC environment. For a stretch of time, I worked on a Windows application and I spent a fair amount of time concerned about Microsoft DLL's, I wrote an article about installation testing and even wrote a blog post entirely focused on DLL hell. DLL hell is an actual term (not just an experience); take a

Continued on page 4

ALSO IN THE NEWS



BUILDING APPS FOR SOCIAL GOOD

If you were thinking of designing or building a website, you'd be in luck...
Continued on page 5

BITBEAMBOT - THE TESTING ROBOT

Can your robot play Angry Birds? On an iPhone? Mine can. I call it...
Continued on page 14

7 CHANGES SCRUM HAS MADE TO TESTING

In the last decade, Agile methods went from quirky and novel to...
Continued on page 29

THE EVIL TESTER QUESTION TIME

More provocative advice for testers who don't know what to do!
Continued on page 32

Main story continued from page 1

Testing in Production (aka TiP)

Software services such as Gmail, Facebook, and Bing have become an everyday part of the lives of millions of users. They are all considered software services because:

- Users do not (or do not have to) install desktop applications to use them
- The software provider controls when upgrades are deployed and features are exposed to users
- The provider also has visibility into the data center running the service, granting access to system data, diagnostics, and even user data subject to privacy policies.

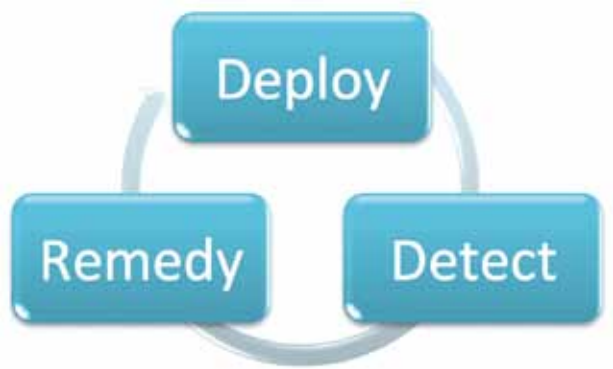


Figure 1. Services benefit from a virtuous cycle which enables responsiveness

It is these very features of the service that enable us to TiP. As Figure 1 shows, if software engineers can monitor production then they can detect problems before or contemporaneously to when the first user effects manifest. They can then create and test a remedy to the problem, then deploy it before significant impact from the problem occurs. When we TiP, we are deploying the new and “dangerous” system under test (SUT) to production. The cycle in Figure 1 helps mitigate the risk of this approach by limiting the time users are potentially exposed to problems found in the system under test.



Figure 2. Users are weird - They do unanticipated things

But why TiP? Because our current approach of Big Up-Front Testing (BUFT) in a test lab can only be an attempt to approximate the true complexities of your operating environment. One of our skills as testers is to anticipate the edge cases and understand the environments, but in the big wide world users do things even we cannot anticipate (Figure 2) and data Centres are hugely complex systems

AUTHOR PROFILE - SETH ELIOT

Seth Eliot is Senior Knowledge Engineer for Microsoft Test Excellence focusing on driving best practices for services and cloud development and testing across the company. He previously was Senior Test Manager, most recently for the team solving exabyte storage and data processing challenges for Bing, and before that enabling developers to innovate by testing new ideas quickly with users “in production” with the Microsoft Experimentation Platform (<http://exp-platform.com>). Testing in Production (TiP), software processes, cloud computing, and other topics are ruminated upon at Seth’s blog at http://bit.ly/seth_qa and on Twitter (@setheliot). Prior to Microsoft, Seth applied his experience at delivering high quality software services at Amazon.com where he led the Digital QA team to release Amazon MP3 download, Amazon Instant Video Streaming, and Kindle Services.

unto themselves with interactions between servers, networks, power supplies and cooling systems (Figure 3).



Figure 3. Data Centres are complex

TiP however is not about throwing any untested rubbish at users’ feet. We want to control risk while driving improved quality:

- The virtuous cycle of Figure 1 limits user impact by enabling fast response to problems.
- Up-Front Testing (UFT) is still important - just not “Big” Up-Front Testing (BUFT). Up-front test the right amount - but no more. While there are plenty of scenarios we can test well in a lab, we should not enter the realm of diminishing returns by trying to simulate all of production in the lab. (Figure 4).
- For some TiP methodologies we can reduce risk by reducing the exposure of the new code under test. This technique is called “Exposure Control” and limits risk by limiting the user base potentially impacted by the new code.

TiP Methodologies

As an emerging trend, TiP is still new and the nomenclature and taxonomy are far from finalised.

But in working with teams at Microsoft, as well as reviewing the publically available literature on practices at other companies, 11 TiP methodologies have been identified (Table 1).

Examples of TiP Methodologies in Action

To bring these methodologies to life, let’s delve into some of them with examples.

Experimentation for Design and Controlled Test Flight are both variations of “Controlled Online

Experimentation”, sometimes known as “A/B Testing”. Experimentation for Design is the most commonly known whereby changes to the user experience such as different messaging, layout, or controls are launched to a limited number of unsuspecting users, and measurements from both the exposed users and the un-exposed (control) users are collected. These measurements are then analysed to determine whether the new proposed change is good or not. Both Bing and Google make extensive use of this methodology. Eric Schmidt, former Google CEO reveals, “We do these 1% launches where we float something out and measure that. We can dice and slice in any way you can possibly fathom.”ⁱ

Controlled Test Flight is almost the same thing, but instead of testing a new user experience the next “dangerous” version of the service is tested versus the tried and true one already in production. Often both methodologies are executed at the same time, assessing both user impact and quality of the new change. For example Facebook looks at not only user behaviour (e.g., the percentage of users who engage with a Facebook feature), but also error logs, load and memory when they roll out code in several stages:ⁱⁱ

1. internal release
2. small external release
3. full external release

Continued on page 3



Figure 4. Value spectrum from No Up-Front Testing (UFT) to Big Up-Front Testing (BUFT)



Tip to myself: Pradeep, test often, your own ideas and conclusions about what good testing is. You would gain significant advantage over lots of other testers just by doing that because they hardly do it. - Pradeep Soundararajan

METHODOLOGY	DESCRIPTION
Ramped Deployment	Launching new software by first exposing it to subset of users then steadily increasing user exposure. Purpose is to deploy, may include assessment. Users may be hand-picked or aware they are testing a new system.
Controlled Test Flight	Parallel deployment of new code and old with random unbiased assignment of unaware users to each. Purpose is to assess quality of new code, then may deploy. May be part of ramped deployment.
Experimentation for Design	Parallel deployment of new user experience with old one. Former is usually well tested prior to experiment. Random unbiased assignment of unaware users to each. Purpose is to assess business impact of new experience.
Dogfood/Beta	User-aware participation in using new code. Often by invitation. Feedback may include telemetry, but is often manual/asynchronous.
Synthetic Test in Production	Functional test cases using synthetic data and usually at API level, executing against in-production systems. "Write once, test anywhere" is preferred: same test can run in test environment and production. Synthetic tests in production may make use of production monitors/diagnostics to assess pass/fail.
Load/Capacity Test in Production	Injecting synthetic load onto production systems, usually on top of existing real-user load, to assess systems capacity. Requires careful (often automated) monitoring of SUT and back-off mechanisms
Outside-in load / performance testing	Synthetic load injected at (or close to) same point of origin as user load from distributed sources. End to End performance, which will include one or more cycles from user to SUT and back to the user again, is measured.
User Scenario Execution	End-to-end user scenarios executed against live production system from (or close to) same point of origin as user-originated scenarios. Results then assessed for pass/fail. May also include manual testing.
Data Mining	Test cases search through real user data looking for specific scenarios. Those that fail their specified oracle are filed as bugs (sometimes in real-time).
Destructive Testing	Injecting faults into production systems (services, servers, and network) to validate service continuity in the event of a real fault.
Production Validation	Monitors in production check continuously (or on deployment) for file compatibility, connection health, certificate installation and validity, content freshness, etc.

Table 1. TiP Methodologies Defined

Continued from page 2

Testing a release internally like this can also be considered part of Dogfood TiP methodology.

Controlled Test Flight can also be enabled via a TiP technique called Shadowing where new code is exposed to users, but users are not exposed to code. An example of this approach was illustrated when Google first tested Google Talk. The presence status indicator presented a challenge for testing as the expected scale was billions of packets per day. Without seeing it or knowing it, users of Orkut (a Google product) triggered presence status changes in back-end servers where engineers could assess the health and quality of that system. This approach also utilized the TiP technique Exposure Control, as initially only 1% of Orkut page views triggered the presence status changes, which was then slowly ramped up.ⁱⁱⁱ

As described, Destructive Testing, which is the killing of services and servers running your production software, might sound like a recipe for disaster. But the random and unexpected occurrence of such faults is a certainty in any

service of substantial scale. In one year, Google expects to see 20 rack failures, three router failures and 1000s of server failures.^{iv} So if these failures are sure to occur, it is the tester's duty to assure the service can handle them when they do.

A good example of such testing is Netflix's Simian Army. It started with their "Chaos Monkey", a script deployed to randomly kill instances and services within their production architecture. "The name comes from the idea of unleashing a wild monkey with a weapon in your data center (or cloud region) to randomly shoot down instances and chew through cables." Then they took the concept further with other jobs with other destructive goals. Latency Monkey induces artificial delays, Conformity Monkey finds instances that don't adhere to best practices and shuts them down, Janitor Monkey searches for unused resources and disposes of them.^v

Synthetic Tests in Production may be more familiar to the tester new to TiP. It would seem to be just running the tests we've always run, but against production systems. But in production we need to be careful to limit the impact on actual users. The freedoms we enjoy in the

test lab are more restricted in production. Proper Test Data Handling is essential in TiP. Real user data should not be modified, while synthetic data must be identified and handled in such a way as to not contaminate production data. Also unlike the test lab, we cannot depend on "clean" starting points for the systems under test and their environments. The Microsoft Exchange team faced the challenge of copying their large, complex, business class enterprise product to the cloud to run as a service, while continuing to support their enterprise "shrink-wrap" product. For the enterprise product they had 70,000 automated test cases running on a 5,000 machine test lab. Their solution was to:

- Re-engineer their test automation, adding another level of abstraction to separate the tests from the machine and environment they run on
- Create a TiP framework running on Microsoft Azure to run the tests

This way the same test can be run in the lab to test the enterprise edition and run in the cloud to test the hosted service version. By leveraging the elasticity of the cloud, the team is able to:

- Run tests continuously, not just at deployment.
- Use parallelization to run thousands of tests per run.

Data is collected and displayed in scorecards, providing continuous evaluation of quality and service availability.^{vi}

Testing Somewhere Dangerous

Production has been traditionally off-limits to testers. The dangers of disruption to actual users should not be under-estimated. But by using sound TiP methodologies, we can limit risk and reap the benefits of testing in production to improve quality, which is ultimately the best way to benefit our customers. □

REFERENCES

- i How Google Fuels Its Idea Factory, Businessweek, April 29, 2008; http://www.businessweek.com/magazine/content/08_19/b4083054277984.htm
- ii FrameThink ; How Facebook Ships Code <http://framethink.wordpress.com/2011/01/17/how-facebook-ships-code/>
- iii Google Talk, June 2007 @9:00; <http://video.google.com/videoplay?docid=6202268628085731280>
- iv Jeff Dean, Google IO Conference 2008, via Stephen Shankland, CNET http://news.cnet.com/8301-10784_3-9955184-7.html
- v The Netflix Simian Army; July 2011; <http://techblog.netflix.com/2011/07/netflix-simian-army.html>
- vi Experiences of Test Automation; Dorothy Graham (soon to be published book: <http://www.dorothygraham.co.uk/automationExperiences/index.html>), Chapter: "Moving to the Cloud: The Evolution of TiP, Continuous Regression Testing in Production"; Ken Johnston, Felix Deschamps

AUTHOR PROFILE - KAREN N. JOHNSON

Karen N. Johnson is a software test consultant. She is frequent speaker at conferences. Karen is a contributing author to the book, Beautiful Testing by O'Reilly publishers. She has published numerous articles and blogs about her experiences with software testing. She is the co-founder of the WREST workshop, more information on WREST can be found at: <http://www.wrestworkshop.com> / Visit her website at: <http://www.karennjohnson.com>

Second story continued from page 1

look on Wikipedia for a good description of DLL hell.

The point of my digression to DLL hell is we can learn from the past. In short, DLL versions can conflict, and version control issues can be encountered on first installations as well as subsequent upgrades of software. Issues arise when different versions of a needed DLL are loaded into memory result in havoc. DLL hell is messy. I recall spending hours tracing DLL versions and investigating and detecting version issues across a series of commonly-used DLLs.

Now with mobile devices we're in a new environment but like most computing environments, how long do we believe our environment will remain pristine as we download applications, install widgets, add games and install assorted applications on our portable little computers? It's time to acknowledge our phones are computers and not just devices. I know I'm currently carrying a phone that has a faster processor speed and more memory than the first laptop I bought. Our phones are powerful. And we are increasingly spending time using applications on our phones.

As testers, it's time once again to think about our computing "environment." I use the term pristine environment to encapsulate thinking of a computing environment as clean, untouched – where no installation has gone before – analogous to fresh snow on the ground. But as we know, the land does not remain untouched for long. We take pictures (and fill the phone memory card). We install fun little games we might never install on our laptop computers and in doing so we alter the very computing environment we are using. Somehow on a PC, these installations are much more noticeable – we might need to insert a CD or if we download an application directly from the web, we see message boxes asking our permission to make changes. We are aware (hopefully) that we are changing the environment on our PCs when we perform these activities. But we are not so aware we are altering the environment on our phones as we snap photos and synchronize calendars and store contact data.

I propose in the coming months and years, application and permission conflicts as well as the age-old issues of processor speed and memory will return as variables to be mindful of as our phones become a primary computing environment. We need to think about testing on devices that are "pristine" as well as on devices that are in fact more "normal" meaning the opposite of pristine. The latter are devices where many applications are loaded, the memory card is nearly full and data synchronization is happening every few minutes or even seconds, GPS software is continually chugging in the background and connectivity is jumping from network to

network as we move about expecting everything to work seamlessly and flawlessly. There are numerous variables to consider – a tester's paradise and nightmare at the same time.

So where do we begin? Start with learning what our mobile applications include. What are the contents of the installation package? By this, I mean file contents and dependencies our mobile application may have; for example, a dependency might be that space is available on a memory card. This is the type of test condition I would test with one or just a couple of devices to see how an application handles a space limit or lack of space entirely (by removing a memory card). Think through and be aware of a mobile application's dependencies such as the chronic need for Wi-Fi connectivity, and test "what happens if" these dependencies cannot be fulfilled.

It's important to know what permissions our mobile applications require. Let's think through an example – imagine we have a mobile application that helps users find store locations and that very functionality – find location – may be dependent on GPS being turned on. What if the user accepts the permission request to use location services but does not actually have location services or GPS turned on? What happens when the user attempts to access the find location feature? What if location services are turned on during the installation of a mobile app but are later disabled?

Thinking through installation and permission dependencies is different than thinking about functional testing conditions. In functional testing, I focus on finding conditions that challenge the logic of an application, whereas in installation, interface and permission testing I'm thinking about how my application interacts with other software or other outside dependencies (such as device settings).

We can download a mobile app incredibly quickly compared to the olden days of client server applications, and yet we cannot easily return to the same test environment because, once an application has been downloaded, we have altered the environment. Since the state of a test environment can be changed so quickly, we should think through these conditions carefully and then mindfully step through these test scenarios. To go back to my fresh snow analogy, we can only walk down an untouched snowy path once before the condition has been changed. I suspect that someday soon, we will have software that allows us to restore a mobile environment (beyond a factory reset) much like we have had software such as Ghost that enabled testers to restore test environments relatively painlessly, quickly and yet thoughtfully.

As a tester, a question to consider is: What are the points of intersection between your app and other apps or other native phone features? Does this sound like interface/interaction testing? This

is another form of testing that returns in the mobile environment. It is so easy for me to become entirely focused on testing an application that I forget about the rest of the world, but in mobile testing, while I'm focused on a mobile app, the phone itself may receive a phone call and that incoming call can quickly change the state of the environment. Yes, incoming phone calls and text messages disrupt my train of thought but do these activities disrupt the mobile application I am testing? Can I multitask? Can my application multitask? Can I resume what I was doing with a mobile app when I'm done using the phone for the old-fashioned need of making or receiving a phone call?

I suggest, at least at times, we put aside our concerns about having hundreds of phone devices on which to test and instead test some conditions at least once or a few times on a small most-frequently used set of devices so that we know the answer to an assortment of "what if this happens" conditions. After all, if an application crashes or misbehaves badly enough on a primary environment, we can report the test condition and resulting issue and move onto our next test condition. It may depend on the type of application you are testing or the team you are working on or your own responsibilities within your team, but I would rather test a greater variety of test conditions on a small set of devices than test fewer conditions on a larger variety of devices.

As often is the case with software testing, there are more conditions to test than there may ever be time in the day when the next thing you know it's time to test the next release or the operating system or another essential variable has made a change. The same is true in the world of mobile; there are firmware upgrades, device upgrades and our own software applications upgrades. Installation testing expands from testing a first-time install versus testing an upgrade. I recently encountered a critical defect when I upgraded a mobile application. So I went "backwards" by removing the software entirely, then downloaded and installed the latest version of the app without going through the upgrade process – the defect did not occur. Some additional testing time and investigation proved the issue only occurred when the app was upgraded versus a first time install.

The good news for a long-time tester like me is I have experience in install, upgrade and interface/interaction testing and I can rapidly bring this background with me in a new frontier – the mobile environment. For testers who are new to the field, this is an opportunity to learn from history as we move forward to yet another new computing platform, the mobile environment. □

REFERENCES

1. Better Software September 2007. "Navigating the Installation: A Charter for Effective Install Testing" by Karen N. Johnson - <http://karennicolejohnson.com/wp-content/uploads/2009/02/kjohnson-better-software-fall-2007-navigating-the-install.pdf>
2. DLL Hell from Karen N. Johnson's blog on the Gold CD see: <http://www.testingreflections.com/node/view/6475>
3. Ghost software: [http://en.wikipedia.org/wiki/Ghost_\(software\)](http://en.wikipedia.org/wiki/Ghost_(software))



Calling all Irish testers to visit www.softtest.ie for Irish testing industry news.



Building mobile applications for social good

By Ken Banks

If you were thinking of designing or building a website, you'd be in luck. If you were thinking of writing a suite of financial management software, you'd be in luck. If you were even thinking of creating the next big video game, you'd be in luck. Visit any good bookstore and the selection of self-help books and "how-to" guides leave you spoilt for choice. People have been working on these things for ages, and good and bad practice in website, financial software or games development – among many others – is well established. The biggest challenge you'd likely face is deciding which book to choose. If you're anything like me you'll leave the store with at least a couple.

Unlike the plethora of self-help guides on the more established topics, if you were looking to do something with mobile phones you'd likely have mixed results. There are plenty of books available extolling the virtues of Java, Python, Ruby, Ruby on Rails, C++, Symbian, Android and just about any other development environment or platform out there. Combine that with the growing field of mobile UI (user interface) design and you'd think that pretty much everything was covered. But there is one thing missing, although you'd probably only notice if you're one of a growing number of developers turning their attention to the developing world.

Building software aimed at helping people solve problems in the developing world is something we've spent the best part of the last six years doing. It's been a particularly exciting time in mobile, with phones working their way into the hands of rural communities the world over, many of whom previously had no method of electronic communication. The opportunities this has opened up have not gone unnoticed, and the non-profit community as much as the commercial world have been keen to understand and take advantage. In this article I'd like to share a few of the lessons we've learnt as part of our journey, starting with a few myths and misconceptions, all of which I believe need busting.

“High-end is better than low-end”

Firstly, one mobile tool should never be described as being better than the other – it's all about the context of the user. There is just as much need for a \$1 million server-based, high bandwidth mobile-web solution as there is for a low-cost, SMS-only PC-based tool. Both are valid. Solutions are needed all the way along the “long tail“ (<http://www.kiwan-ja.net/blog/2009/01/a-glimpse-into-social-mobiles-long-tail/>), and users need a healthy applications ecosystem to dip into, wherever and whenever they may be. Generally speaking there is no such thing as a bad tool, just an inappropriate one.

“Don’t bother if it doesn’t scale”

Just because a particular solution won't ramp-up to run an international mobile campaign, or health care for an entire nation, does not make it irrelevant. Just as a long tail solution might likely never run a high-end project, expensive and technically complex solutions would likely fail to downscale enough to run a small rural communications network. Let's not forget that a small deployment which helps just a dozen people is significant to those dozen people and their families.

“Centralised is better than distributed”

Not everything needs to run on a mega-server housed in the capital city and accessed through the cloud. Okay, storing data and even running applications – remotely – might be wonderful technologically, but it's not so great if you have a patchy internet connection, if you have a connection at all. For most users, centralised means “remote”, while distributed means “local”.

“Big is beautiful”

Sadly there's a general tendency to take a small-scale solution that works and then try to make

Continued on page 6



Continued from page 5

a really big version of it. One large instance of a tool is not necessarily better than hundreds of smaller instances. If a small clinic finds a tool to help deliver health care more effectively to two hundred people, why not simply get the same tool into a thousand clinics? Scaling a tool changes its DNA, sometimes to such an extent that everything that was originally good about it is lost. Instead, replication is what's needed.

"Tools are sold as seen"

I would argue that everything we see in the social mobile applications ecosystem today is "work in progress", and it will likely remain that way for some time. The debate around the pros and cons of different tools needs to be a constructive one – based on a work in progress mentality – and one which positively feeds back into the development cycle.

"Collaborate or die"

Although collaboration is a wonderful concept, it doesn't come without its challenges – politics, ego and vested interests among them. There are moves to make the social mobile space more collaborative, but this is easier said than done. 2011 will determine whether or not true non-competitive collaboration is possible, and between who. The more meaningful collaborations will be organic, based on needs out in the field, not those formed out of convenience.

"Appropriate technologies are poor people's technologies"

A criticism often aimed more broadly at the appropriate technology movement, locally-powered, simple low-tech-based responses should not be regarded as second best to their fancier high-tech 'Western' cousins. A cheap, low-spec handset with five days standby time is far more appropriate than an iPhone if you don't live anywhere near a mains outlet.

"No news is bad news"

For every headline-grabbing mobile project, there are hundreds – if not thousands – which never make the news. Progress and adoption of tools will be slow and gradual, and project case studies will bubble up to the surface over time. No single person in the mobile space has a handle on everything that's going on out there.

"Over-promotion is just hype"

Mobile tools will only be adopted when users get to hear about them, understand them and have easy access to them. One of the biggest challenges in the social mobile space is outreach and promotion, and we need to take advantage of every opportunity to get news on available solutions – and successful deployments – right down to the grassroots. It is our moral duty to do this, as it is to help with the adoption of those tools that clearly work and improve people's lives.

AUTHOR PROFILE - KEN BANKS

Ken Banks, founder of kiwanja.net, devotes himself to the application of mobile technology for positive social and environmental change in the developing world, and has spent the last 19 years



working on projects in Africa. His early research resulted in the development of FrontlineSMS, an award-winning text messaging-based field communication system aimed at grassroots non-profit organisations. Ken graduated from Sussex University with honours in Social Anthropology with Development Studies, was awarded a Stanford University Reuters Digital Vision Fellowship in 2006, and named a Pop!Tech Social Innovation Fellow in 2008. In 2009 he was named a Laureate of the Tech Awards, an international awards program which honours innovators from around the world who are applying technology to benefit humanity. He was named a National Geographic Emerging Explorer in May 2010 and an Ashoka Fellow in 2011, and is the recipient of the 2011 Pizzigati Prize for Software in the Public Interest. Ken was also a member of the UK Prime Minister's delegation to Africa in July 2011. His work was initially supported by the MacArthur Foundation, and he is the current recipient of grants from the Open Society Institute, Rockefeller Foundation, HIVOS, the Omidyar Network and the Hewlett Foundation. Further details of Ken's wider work are available on his website at www.kiwanja.net

"Competition is healthy"

In a commercial environment – yes – but saving or improving lives should never be competitive. If there's one thing that mobile-for-development practitioners can learn from the wider development and ICT4D community, it's this.

So, you've come up with the next big idea to cure malaria or solve the global food crisis. What next? Historically many developers have shown a general tendency to dive straight into programming, but in reality this is one of the last things you should be doing. Here are a few tips we've collected over the years on how to best go about validating your idea, and then how to best go about developing it (should you still decide to).

Firstly, think carefully if you're about to build a solution to a problem you don't fully understand.

Check to see if any similar tools to the one you want to build already exist and, if they do, consider partnering up. Despite the rhetoric, all too often people end up reinventing the wheel.

Be flexible enough in your approach to allow for changing circumstances, ideas and feedback. Don't set out with too many fixed parameters if you can help it.

From the outset, try to build something that's easy enough to use without the need for user training or a complex manual, and something which new users can easily and effortlessly replicate once news of your application begins to spread.

Think about rapid prototyping. Don't spend too much time waiting to build the perfect solution, but instead get something out there quickly and let reality shape it. This is crucial if the application is to be relevant.

Never let a lack of money stop you. If considerable amounts of funding are required to even get a prototype together, then that's telling you something – your solution is probably overly complex.

Learn to do what you can't afford to pay other people to do. The more design, coding, building, testing and outreach you can do yourself, the better. Stay lean. These tasks can be outsourced later if your solution gains traction and attracts funding. The more you achieve with few resources and the more commitment and initiative you show will increase the chances a donor will be attracted to what you're doing.

Don't be too controlling over the solution. Build an application that is flexible enough to allow users, whoever and wherever they may be, to plant their own personalities on it. No two rural hospitals work the same way, so don't build an application as if they did.

Think about building platforms and tools that contribute to the solution for the users, rather than one which seeks to solve and fix everything for them. Let them be part of it. Think about how your imported solution looks to a local user. Are they a passive recipient of it, or can they take it and turn it into their solution? A sense of local ownership is crucial for success and sustainability.

Ensure that the application can work on the most readily and widely available hardware and network infrastructure. Text messaging solutions aren't big in the social mobile space for nothing. And, for the time being, try to avoid building applications that require any kind of Internet access, unless you want to restrict your target audience from the outset.

Every third party the user needs to speak to in order to implement your solution increases the chances of failure by a considerable margin, particularly if one of those parties is a local mobile operator.

Be realistic about what your application can achieve, and wherever possible look for low hanging fruit. Remember – big is not better, small is beautiful, and focus is king. A solid application that solves one element of a wider problem well is better than an average application that tries to solve everything.

Bear in mind that social mobile solutions need to be affordable, ideally free. Business models,

Continued on page 7



When a virtually flawless application is delivered to a customer, no one says how well tested it was. Development teams will always get the credit. However, if it is delivered with bugs, everyone will wonder who tested it! - bhawin

Continued from page 6

if any, should be built around the use of the application, not the application itself. Easier said than done, so try to engage business studies graduates at universities, many of who are always looking for cool social-change projects to work on.

Leverage what local NGOs (or users) are best at, and what they already have – local knowledge, local context, local language and local trust among local communities. Remember that it's unlikely you will ever understand the problem as much as they do, and that it's always going to be easier to equip them with tools to do the job than it will ever be for you to learn everything they know.

Don't waste time or energy thinking too much about the open sourcing process (if you

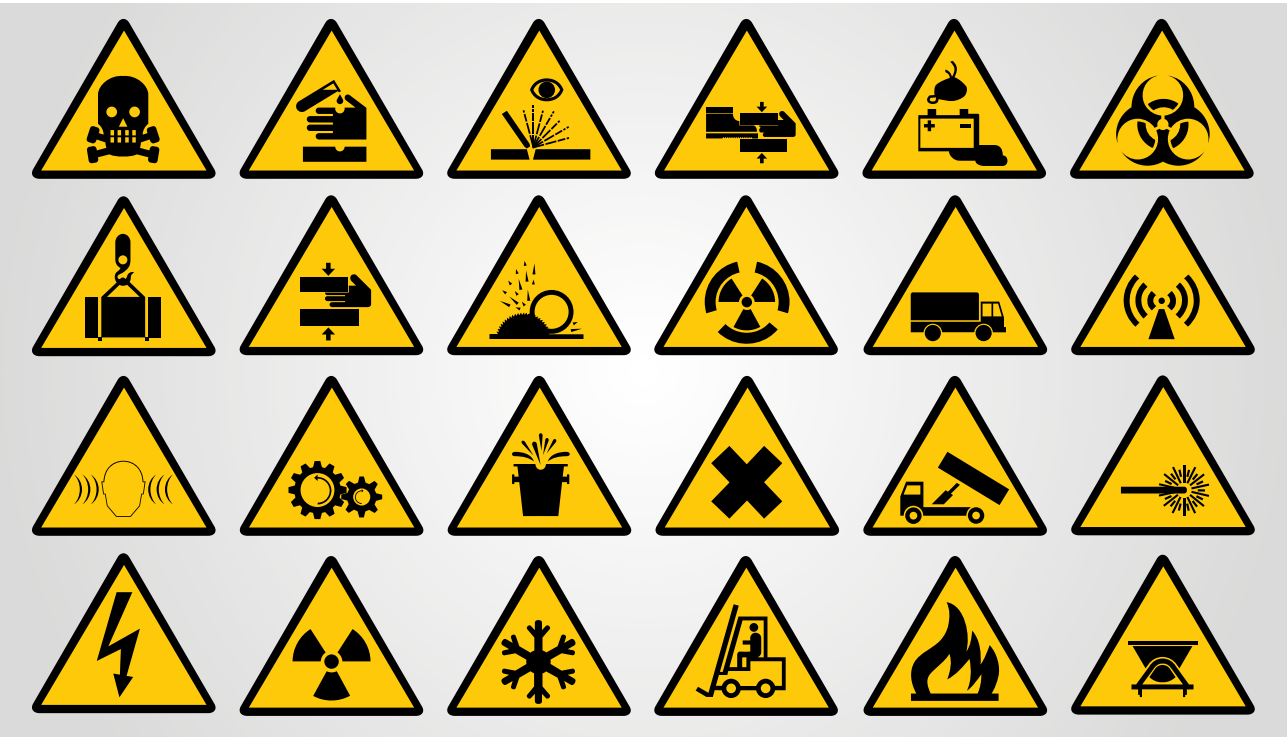
decide to go that route) until you know you have something worth open sourcing. (And, by the way, the users will be the ones to let you know that).

Don't build an application for an environment where it may politically (or otherwise) never be adopted. For example, a nationwide mobile election monitoring system would need government buy-in to be implemented. Governments that commit election fraud to stay in power are unlikely to adopt a technology that gives the game away.

Consider controlling distribution and use of your application at the beginning. Not only is it a good idea to be able to contact users for feedback, donors will almost always want to know where it is being used, by whom and for what. Neglect to collect this data at your peril.

Promote your solution like crazy. Reach out to people working in the same technology circles as you, post messages on relevant blogs, blog about it yourself, build a project website, try and brand your solution, and make use of social networking tools such as Twitter and Facebook. Although your target users may not be present, many are likely to be fairly resourceful, and the more people talking about your solution the more likely news is to filter down to them.

Finally, build a community around the application, encourage users to join and share experiences, and to help each other. Don't be afraid to reach out for additional information, and work hard to keep it active, engaging and growing. Communities are notoriously hard to build, but when they work they're worth it. □



What's at stake?

Advice for managing application security for the next wave of enterprise risk

By Chris Wysopal

Based on Veracode's bi-annual State of Software Security report¹, more than half of all applications submitted for security testing failed to meet acceptable levels of secure coding. And, more than 8 out of 10 web applications fail the OWASP Top 10², an industry standard list of critical web application security errors. So what does this mean? Software security continues to be poor, despite the increasing threat space where application vulnerabilities are being used by attackers to penetrate into corporations to steal intellectual property and proprietary data.

Consider the common denominators across high-profile attacks just this year alone. In the beginning of 2011, SQL Injection was used in

high-profile attacks such as those on the security company HBGary, the Night Dragon attacks on energy companies, and the certificate forging attack on Comodo. SQL Injection vulnerabilities in applications on organization perimeters were leveraged to get insider access to corporate secrets. Another major attack in early 2011 used a memory corruption flaw in a desktop application, Adobe Flash, to bridge the perimeter and install remote access software on an unwitting employee's desktop as a way of penetrating further into RSA and stealing valuable corporate secrets.

These corporate breaches are examples of the two major threat space trends going on today:

1. Attackers are discovering and exploiting common vulnerabilities on web applications to

steal the data they manage or as stepping stones to penetrate deeper into organizations.

2. Attackers are taking advantage of vulnerabilities found in common desktop software purchased by organizations in order to compromise employee workstations as pivot points to get deeper toward their goal.

These trends will continue until solutions are put in place to diminish the quantity of these software vulnerabilities or mitigate their exploitability. Traditional perimeter defence and detection software has been shown to be woefully inadequate against these major threat space trends. We don't need more security software. We need more secure software.

The State of Software Security – And How to Improve It

This poor state of application security can be attributed to a few factors - security processes such as threat modelling or secure coding standards were not incorporated into the development lifecycle, or the security processes were incorporated, but failed to reduce flaws significantly. Another contributing factor is the lack of academic or professional training offered to development and QA professionals on secure coding practices.

In response, some organizations are stepping up to these enhanced attack trends and software security weaknesses by putting in place comprehensive application security programs. These programs seek to instill application security into the software development lifecycle as well as procurement processes that result in the acquisition of commercial or outsourced software.

This article takes a look at one aspect of an application security program – automated software security testing. First, let's study how a security vulnerability can be used to an attacker's advantage.

A Common Vulnerability Illustrated: How SQL Injection Works?

SQL Injection is one of the most commonly exploited vulnerabilities in web applications in particular. It was the culprit behind the recently

Continued on page 8



Continued from page 7

publicized attacks against Sony’s PlayStation Network, for example. Yet, it remains a prevalent vulnerability in web applications being developed today.

Most web applications can really be considered containers to customer or proprietary data. They are designed to provide access to data by authorized users. This access is sought by way of SQL queries that the web application executes against the database. If attackers can influence the SQL that you use to communicate with your database, then they can gain access to data they should not have or escalate their privileges. Consider the following SQL statement, which depicts a typical query that a web application may carry out when a user is logging in to check their username and password:

```
select * from users where username = 'cwysopal'
AND password='r3ally53cur3!'
```

When this query is executed it does a search for the particular username and password in the user table and returns a record if a match is found. Note that the text in red, ‘cwysopal’ is the user name and ‘r3ally53cur3’ is the password that the user supplies by perhaps typing them into a web form. What happens if a user injects data that alters the SQL query in a meaningful way? They are able to do this when they are given free reign to supply any type of user data and relevant protections are not put into the application that restrict that freedom. So, without the relevant protections, a user could alter the SQL query to be the following:

```
select * from users where username = 'admin' AND
password='' OR 'a' = 'a'
```

They supply a username that could be used as the administrator username and supply characters that have special meaning in the SQL language for the password. As it happens the set of characters provided as the password always resolve to True. This has the net result of the user being able to log in as admin.

One traditional approach to preventing SQL Injection attacks is to handle them as an input validation problem and either accept only characters from a whitelist of safe values or identify and escape a blacklist of potentially malicious values. Whitelisting can be a very effective means of enforcing strict input validation rules, but parameterized SQL statements require less maintenance and can offer more guarantees with respect to security. As is almost always the case, blacklisting is riddled with loopholes that make it ineffective at preventing SQL Injection attacks. For example, attackers can:

- Target fields that are not quoted
- Find ways to bypass the need for certain escaped meta-characters
- Use stored procedures to hide the injected meta-characters

Manually escaping characters in input to SQL

queries can help, but it will not make your application secure from SQL Injection attacks. Another solution commonly proposed for dealing with SQL Injection attacks is to use stored procedures. Although stored procedures prevent some types of SQL Injection attacks, they fail to protect against many others.

Automated Security Testing

There are two automated security testing approaches that organizations can employ – static analysis and dynamic analysis. Both are discussed below.

Static Analysis

Static analysis refers to the analysis of software that is performed without actually executing, or running, that software. Static analysis examines applications in a non-runtime environment. This method of testing has distinct advantages in that it can evaluate both web and non-web applications and can detect flaws in the software’s inputs and outputs that cannot be seen by observing the run-time behaviour alone. Many tools that development and QA personnel already employ perform static analysis, in particular, compilers. Examples of static analysis used by compilers include analyses for correctness, such as type checking, and analyses for optimization, which identify valid performance-improving transformations. Static analysis can be performed by analyzing a program’s source code or machine code (oftentimes referred to as binary code or bytecode). Automated security testing technologies utilize static analysis for the purpose of discovering security vulnerabilities in software applications. Some security analysis tools operate on the source code while others operate on the binary code. Some examples of the classes of vulnerabilities discovered by static analysis include SQL Injection, Cross-Site Scripting and Buffer Overflows.

The advantage of static analysis is that it allows for complete code coverage because it is not dependent on discovering an execution path in order to test it as it is not operating against the application in run-time. Binary static analysis extends the code coverage to third-party

components and libraries that are embedded in the final form of an application for which there is no source code available. In this manner it provides more thorough coverage than source code static analysis. However, static analysis cannot offer any insights into vulnerabilities in the underlying configuration or deployment environment of an application.

Dynamic Analysis

Dynamic analysis operates by executing a program and observing the executions. Testing and profiling are standard dynamic analyses. Dynamic analysis is precise because no approximation or abstraction need be done: the analysis can examine the actual, exact run-time behaviour of the program. By virtue of operating against the running form of an application, dynamic analysis most closely mimics how a malicious user would attack the application. For the purpose of security testing, fully automated dynamic analysis is a technique that can be applied to web applications only. By directing a series of requests to the web application and evaluating the responses received, a determination can be made about the presence of commonly occurring vulnerabilities such as SQL Injection or Cross-Site scripting.

Dynamic analysis has the advantage of being able to expose vulnerabilities in the deployment environment. It also does not rely on the existence of source code. However, since it is difficult to discover all possible execution paths through an application it does not provide the complete coverage that static analysis does.

Summary

In summary, because of the threat space trends and new deployment platforms, including mobile, application security continues to be a dynamic arena. In the past, security testing was only contemplated for very high-risk systems such as online banking, but high-profile breaches in the news tell a different story. Attackers are going after any application that has data of seemingly little

Continued on page 9

REFERENCES

1. <http://www.veracode.com/reports/index.html>
2. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

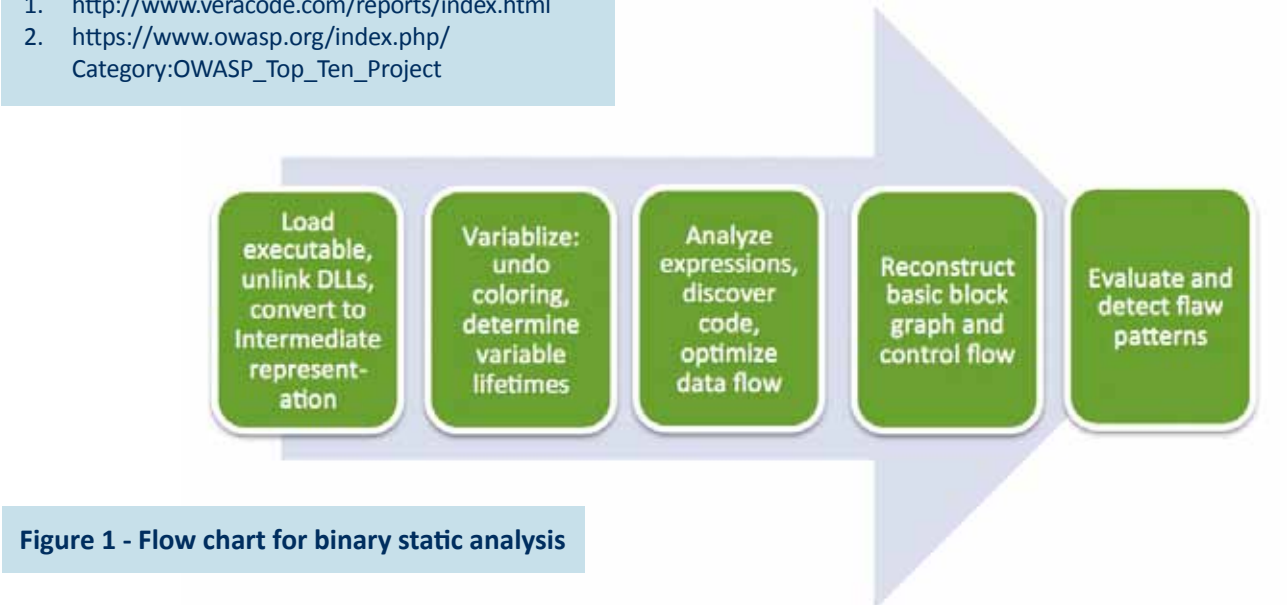


Figure 1 - Flow chart for binary static analysis

Continued from page 8

value to breach corporate networks and tarnish corporate brands. No matter what testing method, or combinations of methods you employ, organizations will need to be nimble and able to scale their programs to manage exponential new applications over many different platforms to keep pace with attackers in the coming years. □

AUTHOR PROFILE - CHRIS WYSOPAL

As co-founder, CTO and CISO of Veracode, Chris Wysopal is responsible for the security analysis capabilities of Veracode technology. He's a recognized expert and well-known speaker in the information security field. Wysopal's groundbreaking work while at the company @stake was instrumental in developing industry guidelines for responsibly disclosing software security vulnerabilities. He is a founder of Organization for Internet Safety (OIS), which established industry standards for the responsible disclosure of Internet security vulnerabilities. Wysopal is co-author of the award-winning password auditing and recovery application @stake LC (L0phtCrack), currently used by more than 6,000 government, military and corporate organizations worldwide. He is also the author of "The Art of Software Security Testing" published by Addison-Wesley.

**Software-Testing-Europe.com**



Find the
best software
testing positions !

JOB board, Training, News for professional testers



The growing cyber threat and the tester's call to arms

By Mark Crowther

I talk to a lot of testers about where they should focus their professional development. There's no simple answer of course— it's a personal choice but it's still a good conversation to have. Given our profession is so broad and individuals so different, a lot of ideas come up. When discussing what particular areas testers might focus on, three seem to come up repeatedly: the broad field of functional / system testing and the more specific areas of performance and security testing.

Upon further questioning, the conversation seems to take on a very similar pattern. Testers say they've got a good level of experience in functional / system testing, have 'done a bit' of performance testing and almost always leave security 'to the professionals' who know how it's done.

Even when talking to owners of other

small to medium sized testing consultancies like Test Hats, the story can be the same. They'll of course state that system testing is trivial for them to deliver, talk up capability and experience around performance or in some cases say they've 'done a bit' of performance testing, which is a little worrying. Then when it comes to security testing, they either say they've not touched on it (generally no 'done a bit' for security thankfully) or again they claim to have it in hand, which I rarely believe.

I've spoken about security testing to founders of three SME sized consultancies who said they deliver it, asking what they provide. As before, there was a lack of clarity over the different levels of security testing or hints about how different forms of security testing get delivered.

Just like performance testing, this appears to be a case of 'doing a bit' of security. Have I used the word 'worrying' yet?

Stand in your corners

The current state of affairs for how the majority of security testing gets delivered appears to be through consultancies that specialise in the field. (If you just mentally said 'yep...', thank you)

However, they tend to not do all the other testing or provide related services that a regular testing consultancy would. Of course there are some huge service companies out there that are exceptions, but in the main we have the 'regular' testing profession on one side and then on the other side we have the security testing consultancies. It's always struck me as an odd disconnect from our side, but perhaps understandable from theirs. Security is complex and it's much easier to achieve the required competence and competitiveness if you specialise.

How many projects have we worked on where security testing consists of bringing in a third party to do Pen Testing? As any astute tester will realise, this approach isn't good practice in terms of securing the application or the network it sits on. Certainly it's of no use in making sure both remain secure, perhaps in a way some external authority requires.

Let's just summarise the three glaring issues we've already mentioned.

- There's a worrying level of ignorance over security testing amongst testing professionals and testing consultancies.
- Security is often bolted-on at the end of the project to assure the application is secure, in a way that's similar to doing testing at the end to bolt-on quality. It doesn't work in either case.
- Professionals with the same essential goals are often separated project wise and professionally.

Ignorance is not bliss

As testing professionals we're on the front line

Continued on page 10

Continued from page 9

of assuring the quality and correctness of the applications we're charged with testing. That means more than just ticking requirements boxes, as seems to be the accepted norm these days.

However, while we might consider getting involved in conversations about defining and clarifying functionality, compatibility, usability, etc. how often does the conversation extend to application security?

When was the last time you got involved in a Threat Modelling session at the design stage of an application you'd later have to test? When was the last time you even heard of this taking place? It's rare, so don't think it's your oversight if you haven't picked up on it happening on your projects.

Here's an example of another risk that ignorance of the need to consider security brings. No threat modelling suggests an assumption that a secure application will just emerge.

This lack of consideration about security testing has already introduced two risks,

- Not considering how to make the application secure at the design stage Expecting it to become or be proven (magically) secure at the end of development.

These are the same issues we still see too often with testing and QA, pulling in regular testers or security testers at the end of development. Yes, it still happens even with all the talk of agile.

Unlike regular testing though, there is an even bigger risk that not considering security can bring. Whereas a website with bad functionality might result in a loss of sales or users for the site alone, an insecure site can risk the privacy and finances of everyone who ever visits the site, and the business will suffer a reputation hit just to compound the situation. Add to that the risk of non-compliance to standards such as PCI and the potential for fines. Clearly, security is not something to be lightly considered.

The growing Cyber threat

Each year the cyber threat increases – with more and more sites being hacked and personal or proprietary information being stolen. This cyber threat applies to individuals, companies and nation states alike. In 2010 the US Secret Service arrested 1200 people in relation to cybercrime. These individuals were located across the US and Europe.

Hactivist groups like Anonymous have attacked many targets and released personal details of hundreds of their customers. From Sony, Apple and BART to PayPal, MasterCard and Visa any company website can be attacked and data compromised. Add to that the activities of Lulzsec and Team Poison (TeaMp0isoN) and sites like RankMyHack.com encouraging hacking, it's clear the problem is not going away.

It's not just websites that are under threat either, as we learned in the McAfee report this year. [1] Governments around the world have been subject to hacker attacks, along with companies in construction, energy, electronics, news, defence,

AUTHOR PROFILE - MARK CROWTHER

Mark Crowther is the founder and Principle Test Architect at Test Hats. His focus is on doing hands-on system and security testing and helping others learn how. You can follow Mark on Twitter at @MarkCTest.

accounting, insurance, sport and more. The UK Cabinet Office estimates the cost of cybercrime to the UK economy to be £27bn, incredible when you consider the cost from drug crime is said to be £13bn. [2]

Just in case you thought you weren't involved, the Damballa half year report for 2011 informed us that the number of systems running Botnets [3] for malicious purposes on the internet, was a staggering 9.2 million just in North America. [4] Not only is the matter of needing secure software not going away, the attacks are increasing and the risks are getting more diverse while increasing in number [5]. If you look over the Web Hacking Incident Database (WHID) [6] and compare it against the OWASP Top 10 [7], you'll see there are far more attack types than just the 10 we're used to hearing about.

SQL injection, which is a basic method of hacking, still remains the number one way to attack a site. In 2010 it accounted for 15% of all recorded attacks in the WHID and ranked number 1 on the OWASP list.

Remember, with the WHID not every attack makes it to the database as not every attack or breach gets disclosed. As with other sources these numbers have to be taken as indicative. Perhaps more worrying than the 11% of attacks using stolen credentials, 6.7% using Cross Site Scripting or 5.3% using good old fashioned Brute Force attacks, is the massive 21% of an unknown attack type. Either there is a serious classification problem or some very sophisticated attacks are happening. The point is clearly made - the problem is not going away and it is becoming more widespread.

Proceed to the door marked Security

The message running through all of this is that there's a need for robust

security testing of software. As software testers we should be in a strong position to lend our support to the Cyberwar, yet are we playing our part?

If we want to, we can step-up to the front lines. Whilst we are not perhaps the ones who should test across the whole security programme, we can certainly prepare the ground for a more complete assault on the application to root out those dangerous security vulnerabilities.

In fact, it's my view that just as we often bridge the gap between different departments, coercing them into collaboration to help improve software quality, we can also do the same to improve software security. Who's been in the situation as a testing professional where we're bringing together Product Owners and Developers, Operations Teams and Support Teams, all in the name of improving quality? Well, now we can be the ones who do the same by pulling together Security teams and System Architects, in the name of improving software security.

What's more, it's something we really should be paying more attention to given all-of-the above and given our profession's increasing use of exploratory testing. This is the killer technique in our toolbox that we can bring to security testing. As we've stated before here at Test Hats, it's our contention that testers who practice exploratory

Continued on page 11

ElectroMind

YOUR SOFTWARE QUALITY MATTERS



Five reasons to learn the **Rapid Software Testing** methods developed by James Bach and Michael Bolton...

- Test effectively even if specifications are poor
- Save money by eliminating unnecessary work
- Save time with Exploratory Testing
- Learn practical Agile techniques that work

If you spot any defects in this advertisement please email **rst@electromind.com** and receive a £100 discount voucher off the next course when you book and pay online.

Don't delay, book today at...

www.electromind.com



Why was there a bug in the computer? It was looking for a byte to eat. - @steveo1967

Continued from page 10

testing techniques have learned a number of essential skills for testing in a security context. Thinking, observing, questioning, reasoning and testing ideas and insights, applying knowledge and acting on new information as it arises — these are essential skills for a security tester. Those who regularly practice exploratory techniques are well equipped to test in a security context.

First steps into security

What needs to happen is for testers to decide they want to take part in security-related testing and define the level at which they will be involved. It's certainly essential that we take responsible and measured steps, only committing to deliver what we are capable of doing. Perhaps it's helping make sure those Threat Assessment meetings happen and being the facilitator and note taker, or maybe helping ensure the design is created with security in mind. Maybe it's doing an Application Vulnerability Assessment in line with the OWASP Top 10, bearing in mind the results of the Threat Assessment to make sure the testing conducted is relevant.

For those who are now focused on testing management, it could even be a more QA focused approach, rallying the business to develop an Information Security Management System (ISMS) in line with ISO27001. There's a lot we can do in the security space as testing and QA professionals, even where that is just 'facilitating' or laying the groundwork for the more experienced professionals.

It's up to us to bridge the gap and the real challenge is in thinking how to get started. Here are some ideas to start things off:

- Over at the Software Testing Club [www.softwaretestingclub.com], you'll find the Security Testing Group that's free to join. The group has a growing list of resources and links to websites, blogs and security forums.
- Start reading a general security blog such as www.ehacking.net. It's not a community forum but has lots of interesting posts on tools, tutorials, etc.
- Have a look at www.securityweek.com and similar sites to get an idea of security in the broader industry and some of the big issues that are happening.
- Read the OWASP Testing Guide or Open

- Source Security Testing Methodology Manual to get an idea of how to approach security testing in a structured way.
- Practice on a system you are allowed to practice on but be sure to practice, practice, practice.

Be sure to head over to the Software Testing Club's Security group and join in the conversation and share your experiences and ideas. Together we can add security testing to our testers' arsenal and help defend against the growing cyber threat. □

REFERENCES

- 1 <http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf>
- 2 <http://www.cabinetoffice.gov.uk/resource-library/cost-of-cyber-crime>
- 3 <http://searchsecurity.techtarget.com/definition/botnet>
- 4 http://media.scmagazineus.com/documents/28/damballa_threat_report-first_h_6879.pdf
- 5 http://www.cso.com.au/slideshow/397747/10_scariest_hacks_from_black_hat_defcon/
- 6 <http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database>
- 7 https://www.owasp.org/index.php/Top_10_2010-Main



By Anne-Marie Charrett

Do you catch yourself dreaming about working in the best test team ever? Maybe one where you get to adopt any new philosophies you chose? How about one that gives you complete autonomy to try new things without having to justify them in triplicate? Perhaps you feel frustrated by a seeming lack of willingness to embrace change in your current organisation? Or driven mad by the countless myths and misconceptions that your company mistakes for testing? Or are you plain fed up with waiting for that promotion that for whatever reason always seems to be just out of your grasp? A new job at a new company can be extremely appealing especially if you feel frustrated

and fed up at your current workplace. But before you abandon ship, look around one last time, check to see if you really have explored all avenues. You may discover some opportunities that you hadn't previously considered; opportunities for example to learn new skills. Why not try creating a 'dream' job description. You know, that one you really deep down want? Allow yourself to explore and dream a little of what it would be like there. Is it more autonomy, or perhaps its something specific, like wanting to adopt a more Exploratory approach to testing? Be brave and make the dreams big! Next work out how this job differs with where you are at right now. Be honest and make it personal. It's not a test; this is for you only. Work out where the gaps are between what you want

and where you are; this is where you will want to focus your energy. Make a list of things you want to work on. Try and make them as concrete and real as possible by being specific. The more specific and detailed you are, the easier it will be able to try out. Yeah sure, you might not get a chance to try out being a Test Manager, but you could perhaps practice some related skills. By now you may have an idea of the skills you want to improve. The real challenge is to find ways in your current role to learn and apply these skills. To do this, you will need to sell your ideas in such a way that these skills are perceived to benefit either the company or key stakeholders.. Preferably both. Start by looking at the problems

Continued on page 12



Continued from page 11

you and your team are encountering. Speak to people who have a vested interest in testing. This could be your fellow testers and your team leader, or it might be a project manager.

Discover what drives them, what's frustrating them and what they need from your testing to get the job done? What could be done to make life easier for them?

Sometimes when speaking to people, it's easy to mistake a symptom as the problem, so a word of caution here, and take the time to fully understand what I call the "underlying problem". (This itself can take a bit of skill but it's one that's well worth learning). If you spend time trying to solve symptoms, it's strange but the underlying problem never seems to get solved!

Now it's time to work some magic. Look at these problems and examine them. Is there something in all this mess that you can use to advocate on behalf of that new skill, or implement that change? If your test manager is over stretched, why not offer to help out on some outstanding tasks?

And you will need to speak to your boss. Explain your ideas and make it clear that this will benefit the company too. If you're nervous about doing this, seek out people who may be receptive to your ideas. Then you can approach your boss as a collective.

Even better if these people have influence within the organisation. Speak to them and ask them for some ideas on how you might influence some change. Failing that, use the testing community. There are plenty of people willing to mentor and help you through a particular challenge. (For an example of this, go to my podcast on the EuroSTAR website, where I help coach a tester through a particular challenge).

The things I'm suggesting here are not always easy wins. Hopefully you will at least

partially succeed in getting what you aimed for; on the other hand you may come out of this process feeling that you got nowhere.

But that's not quite true is it? Look back at the skills you've had to use so far? In particular I'm talking about being persuasive and influencing people. These are real and incredibly valuable skills to have.

Personally, I think the concept of a dream job is a myth. No matter where you work, there will be challenges to face, both technical and/or people related. If you move around a lot (like me), you'll start to notice that often the same problems appear to move with you, suggesting that perhaps the problem lies within!

Are you sure that by moving to a new role the problems won't reappear, albeit in a slightly different way?

The more I ponder this issue, the more I become convinced that testers need to be able to influence people around us in order to effect change. Very simply, if you want things to change

around you, learn to influence those around you.

Look at it this way, you may even find that by creating the learning opportunities you want, you feel sufficiently compelled to stay where you are. On the other hand, if you still feel like you want to leave after up-skilling, you will have some new skills to put on your resume.

Regardless of the outcome and decision you finally make, you have learned a little about influencing and effecting change within an organisation. Do not overestimate the power of these skills. They are gold dust!

I always like to walk away from a company or job knowing that I've tried my hardest to work in that environment. After all, no work place or boss is perfect.

When it comes to change, it can be easy to let fear gets the better of us. We make assumptions that our companies are adverse to change, or that they will never adopt the ideas we have. That may not necessarily be the case. Maybe they need a brave tester to put ideas on the line, in a persuasive and clear way. Maybe, just maybe, that tester is you. □

AUTHOR PROFILE - ANNE-MARIE CHARRETT

Anne-Marie Charrett is a testing coach and trainer in software testing hell bent on helping testers discover their testing strengths and become the testers they aspire to be. Anne-Marie is working on a book with James Bach about coaching, with a target completion date late next year. Anne-Marie offers free IM Coaching to testers and developers on Skype (id charretts).

Anne-Marie has worked in the testing industry for more years than she let's on. She believes that its time to bring testing out of the dark ages and into a new age of enlightenment, where testers all over the world are admired and respected for their critical thinking and for being, well so darn smart! She also is campaigning to make St Thomas the patron saint of testers, after all, he was a true skeptic.

Anne-Marie describes herself as Irish-Australian having been born in Dublin and now residing in Sydney, Australia. She happily engages in work in both the Northern and Southern Hemisphere. Anne-Marie can be found on twitter at charrett and also blogs at <http://mavericktester.com>.

A look inside Original Software

WHAT'S YOUR NAME, COMPANY NAME AND PRODUCT SET YOU OFFER?

Colin Armitage from Original Software. Our company slogan is 'Delivering Quality Through Innovation' and that's exactly what we offer. Our solution suite embraces the full spectrum of Application Quality Management across a wide range of applications and environments.

Our products include a quality management platform, a manual testing product, a test automation solution and test data management software.

Quality needs to be addressed at all layers of the business application and we enable customers to check every element of the application - from the visual layer, through the underlying service processes and messages, as well as into the database.



Original Software

An interview with Colin Armitage from OS

WHERE ARE YOU BASED?

We are based in the UK in a lovely town called Basingstoke in Hampshire. With our US headquarters in Chicago. Partners and distributors sell our software around the rest of the world.

HOW MANY PEOPLE WORK FOR YOU?

We have a core team of exceptional people, which we augment through our various strategic partnerships.

WHAT ARE YOUR BIGGEST DAILY CHALLENGES?

Being several steps ahead of the competition is both a necessity and a challenge as we continue to disrupt the legacy players like HP, IBM and Micro Focus. Our success to date has been based on our ability to punch above our weight with both our marketing strategy AND technological innovation.

Continued on page 13



Attend or start a local testing meetup - <http://www.meetup.com/SoftwareTestingClub/>

HOW DO YOU SEE THE TESTING WORLD CHANGING IN THE FUTURE?

I think that the testing community will continue to struggle keeping up with the rapid changes in technology. New client capabilities and the re-emerging need to test on multiple platforms (a variety of browsers and devices) will carry on.

The shift to agile methods is also creating a need to be able to test faster, automate as much as possible and deal with constant change. I'm pleased to say that our solution suite enables this increase in productivity for software testing.

WHY IS TEST AUTOMATION IMPORTANT IN ANY DEVELOPMENT ENVIRONMENT?

Companies are always looking to do more with less and test automation is a key way to ease the testing burden, improve quality and enable development and QA teams to improve their productivity.

IS TEST AUTOMATION ONLY IMPORTANT FOR AGILE TEAMS?

Not at all. Obviously test automation in Agile is critical if you are to stay ahead of the game within each iteration, but automation is just as valuable in other environments as well. Without it benefits, such as increased test coverage, reduced business risks, improvements in productivity, faster time to market and a reduction in costs, just wouldn't be possible.

WHY DO YOU THINK THE TESTING MARKET IS SO INTERESTING, AND YET DEMANDING AT THE SAME TIME? (MAYBE YOU DON'T?)

It is a market place that is reaching a tipping point in maturity and the demands placed upon it. We offer a unique solution to bring the often competing demands of cost, time and quality into harmony.

IF YOU COULD CHANGE ONE THING ABOUT THE TESTING WORLD WHAT WOULD IT BE?

To make testers a lot more open to innovation rather than relying on legacy tools that have been around for years! I'm sure you know who I'm referring to here...

WHERE SHOULD PEOPLE GO TO FIND OUT MORE ABOUT YOU AND YOUR COMPANY?

The web is the obvious choice at www.origsoft.com. For those who love Twitter, we can also be followed there as we just love engaging with the testing community in this way. Our Twitter handle is @origsoft. We'll also be at EuroSTAR, so maybe the team will get to meet you in Manchester. □

How to recruit a tester

1

BE KNOWN AND ACTIVE

Be known and active in the testing community and when you have a vacancy then a tweet can attract interest from testers who are thinking working with you would be good. Have a connection with some of the known tester names so they can post details of your vacancy to places where testers are likely to see it, e.g., Yahoo groups. Having the vacancy on your site won't get it seen and posting it to one of the job boards means it will get lost in the mass - unless it happens to be the STC job board.

Be part of the testing community so that when a job is tweeted or blogged or put into a mailing list then you see it - and it also lets you know that the job is with someone who takes the time to be part of the testing community.

2

MAKE SURE THE JOB DESCRIPTION SOUNDS APPEALING

Having got the attention of a tester, make sure the job description sounds appealing. Descriptions such as 'create test cases, log defects, excellent communicator' will get a yawn. Mention of ISEB/ISTQB certification - especially if you are recruiting for an agile role - is also likely to have a negative effect. Make it easy for a candidate to find out about your company and its values. Not some bland mission statement on the website but blogs written by people there so the candidate can get a real feel for what it's like to work there and starts to want to work there.

Have a blog. Join the STC, start discussions and answer them. Try and answer questions on QA Stack Exchange. Have a Twitter account. Take part in Weekend Testing. Let the hirer be able to find out who you are - your online reputation can act as your CV and can help bypass some of the first stages of the recruitment process. Research the company - can you find out about its values and principles? What sort of project and technologies do they use?

3

MAKE THE APPLICATION PROCESS CHALLENGING

Make the application process challenging and make yourself stand out. Instead of asking for a CV, ask the candidate to write a short essay. You want communication skills? Get the candidate to show that they have them.

This part should be a breeze - you're being asked to demonstrate something you're good at.

4

GIVE THEM REAL-LIFE TESTS

Don't ask them how they would test a toaster. Get them to test one of your projects. Don't just give them a tour of the office and then park them in an interview room. Let them see and feel the office atmosphere so they start to get a really good idea of what it would be like to work there.

Be yourself - that's what you'll be if you're made an offer and you accept. Look around and imagine yourself working there every day - does it feel right? What does the atmosphere feel like? You'll already have researched the company (see 2), so you can ask intelligent questions.

On a personal note - thanks to those that presented some good opportunities whilst I was searching - sorry I wasn't able to take you up on them but the opportunity at Atomic Object was just too good. □

AUTHOR PROFILE - PHIL KIRKHAM

Phil was a programmer for more years than he would care to admit before he found his true calling of being a tester. After several years of being a test consultant he is now working as a remote contractor for Atomic Object whilst he waits for his visa. Phil acts as the Terminator for the STC, is a member of the Miagi-Do School of Testing and blogs at <http://expectedresults.blogspot.com>



BitbeamBot

The Angry Birds playing, mobile testing robot

By Jason Huggins

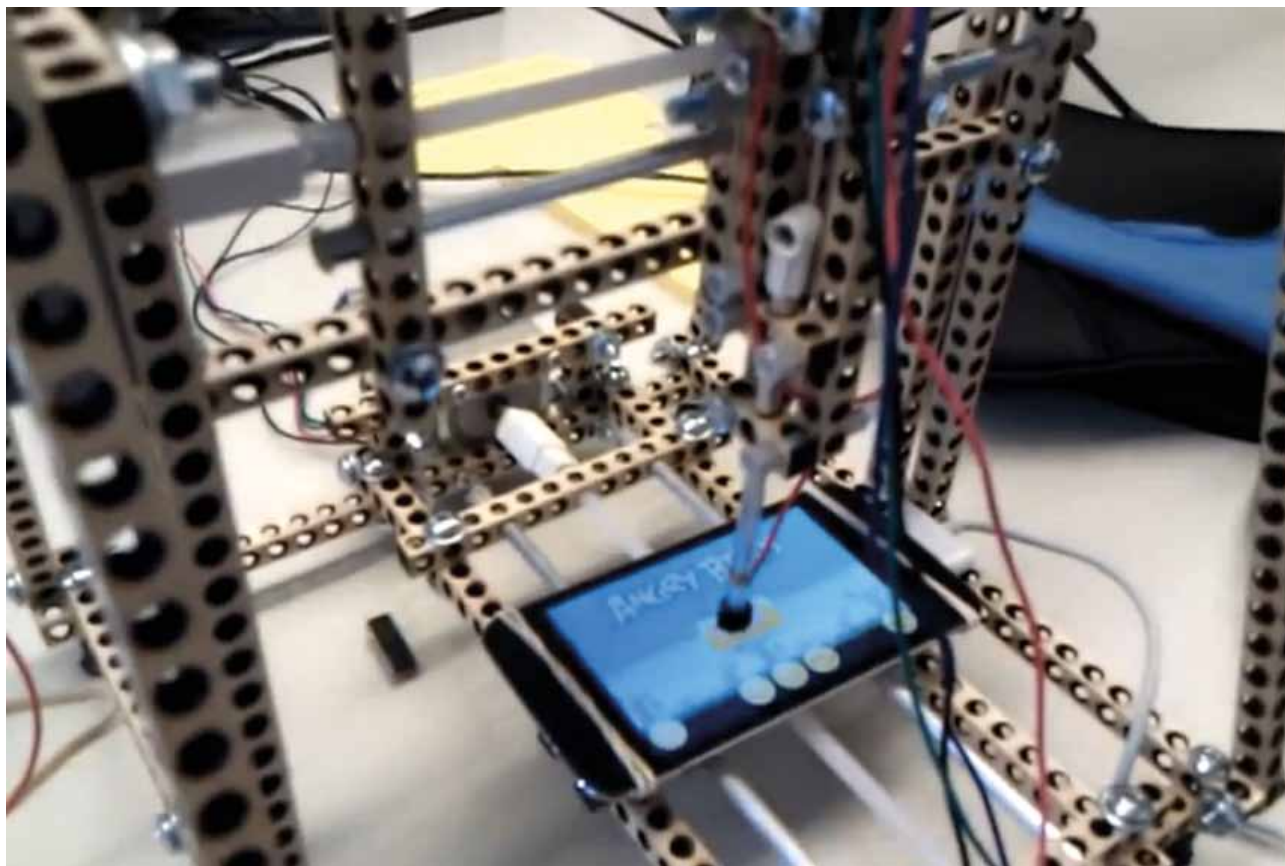
Can your robot play Angry Birds? On an iPhone? Mine can. I call it “BitbeamBot”. It’s a creation that’s been about ten years in the making. As originally conceived, it was just an art project for fun. As I built it and gave it more thought, I came up with a much more practical application: mobile web testing. Let me tell you a bit about how I got here.

About ten years ago, on my desk sat a Pin Art display. Pin Art, or Pinscreen, is a popular desk toy containing an array of pins that you can push your face or hands into and leave an image.

One day when playing with the Pinscreen, I had a thought: what if you could motorize and computer control each individual pin? If you could do that, you could create a display that shows 3D computer graphics on a real honest-to-goodness 3D display. At the time, I was playing around with 3D graphics simulations on my computer and was frustrated by the irony of viewing 3D objects on my very flat 2D computer monitor.

As I saw it, my project involved three big components: software, electronics, and mechanics. Since I had a background in software programming, I tackled the software side first. About 8 years ago, I created a Python and OpenGL-based software simulation. I dubbed it “PinThing”. To make the demo more accessible to more people, last year, I ported the Python version to JavaScript and HTML5 and now host it at pinthing.com. (Source code: <https://github.com/pinthing/pinthing.com>)

With the software side moving along well, it was time to turn my sights to mechanics. After a few failed attempts to make a mechanical prototype with wood and metal, I turned to my

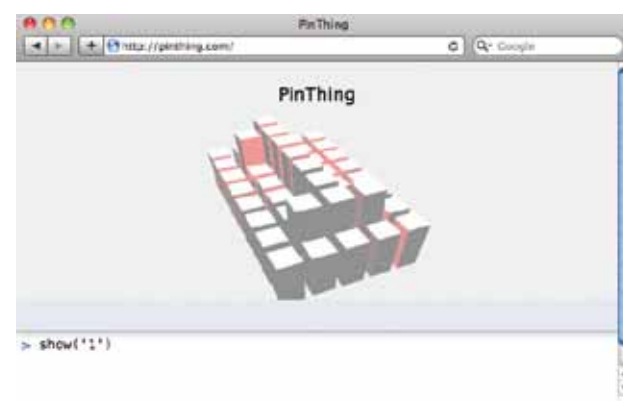


BitbeamBot ready to play Angry Birds

favorite prototyping material, Lego Technic. Lego Technic is like regular Lego bricks, but with holes for attaching wheels, or gears, or pulleys. You can create almost any structure or mechanical device with Lego Technic beams, pegs, and axles. After a few tries, I had a working Lego prototype for one PinThing pin. While I was excited to have a workable design, I was sad when I realized how expensive it would be to buy enough Lego to scale this up to as many pins as needed -- dozens to hundreds of pins for a really big display.

A few days later, I realized the solution to the cost issue - make my own Lego! Of course, that was easier said than done. Fortunately, I had stumbled upon the grid beam project at Maker Faire 2010 in San Mateo, California. Grid beam is a modular design for building things, like tables, chairs, shelves, and even vehicles. Grid beams system is composed of square beams of wood or metal, with holes evenly spaced. (More information about grid beam is available at <http://www.gridbeamnation.com>).

I ported grid beam dimensions down to Lego proportions. I bought pieces of basswood at the local art supply store, and took a class on



PinThing.com

laser cutting at Tech Shop in San Francisco. I laser cut holes into the wood to create what I now call “bitbeams”. (More info at <http://bitbeam.org>)

It then was easy to port over my Lego PinThing design to Bitbeam.

Here’s my first prototype pin made out of Bitbeam. If you look closely, there are a few Lego Technic pieces in there, too.

In my day job as CTO at Sauce Labs, my

Continued on page 15



A Pin Art display



#gtac actually testing is not dead. the end statement from James cleared the air. Test as a Phase is dead. The work still will be done :) - @jhinkoo

Continued from page 14

job is to lead the product direction of our cloud testing service. We have a cloud-based array of desktop browsers like Firefox, IE, and Chrome available for test automators to drive using the Selenium API as an integral part of their continuous integration processes. For a while, we've been planning how and when we'd tackle the problem of mobile testing.

After playing with the PinThing/Bitbeam pin prototype, I noticed something. The mechanics required to move a pin up and down are the same to touch the screen of a mobile phone. If it can touch a screen, it can be used for mobile testing. And if it can be used for testing, this just went from a silly weekend art project to something that might be useful back at work.

But how could a bunch of wooden Lego-like beams be useful to software testing?

For the confidence that your mobile app truly works, you need an end-to-end test on the actual device. This means the full combination of device manufacturer, operating system, data network, and application. And since the devices were meant to be handled with the human hand, you need something like a real hand to do real end-to-end testing. At some point, after lots of repetitive manual testing, the inevitable questions is asked "Can we / should we automate the testing of the old features, so I can focus the manual testing effort on the new features?"

That's where the BitbeamBot comes in. To turn the PinThing into a robotic button clicker I needed to do two more things. First, I added motors to move the pin thing in any direction left, right, forward, or backwards over the screen surface.

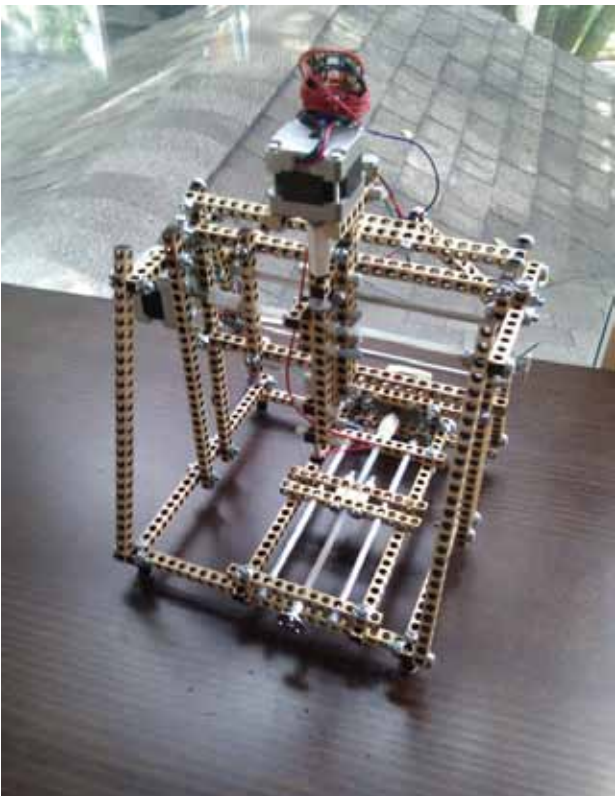
Second, to touch the screen just a like a real finger, I duct taped conductive foam to the end of the pin. (MacGyver would be proud.)

At the moment, BitbeamBot is only a prototype and far from production-ready, but it can play games with simple mechanics, like Angry Birds. However, it's not very smart; it can't yet "see" where objects are on the screen. From my computer, I send electrical signals to two motors to move the pin over any point on an iPhone screen. I then use a third motor to move the pin down to the screen surface and click or drag objects. This open loop, flying blind approach to automation is how automated software testing was done years ago. It's the worst way to automate. Without any sense of what's actually visible on screen, the script will fail when there's a discrepancy between what's actually on the screen and what you expected to be on screen at the time you wrote the automation script.

A better approach to testing with BitbeamBot will involve closing the feedback loop and have software determine where to click based on what is actually on screen. There are two styles I'll experiment with: black box and grey box. Black box testing is easier to get started with, but grey box testing is more stable long term.

Black box testing requires no internal knowledge of the application. It treats the application like a metaphorical "black box". If something is not visible via the user interface, it doesn't exist. To get this approach to work with BitbeamBot, I'll probably place a camera over the phone, send the image to my computer, and use an image-based testing tool like Sikuli. Sikuli works by taking a screenshot and then using the OpenCV image recognition library to find elements like text or buttons in the screenshot.

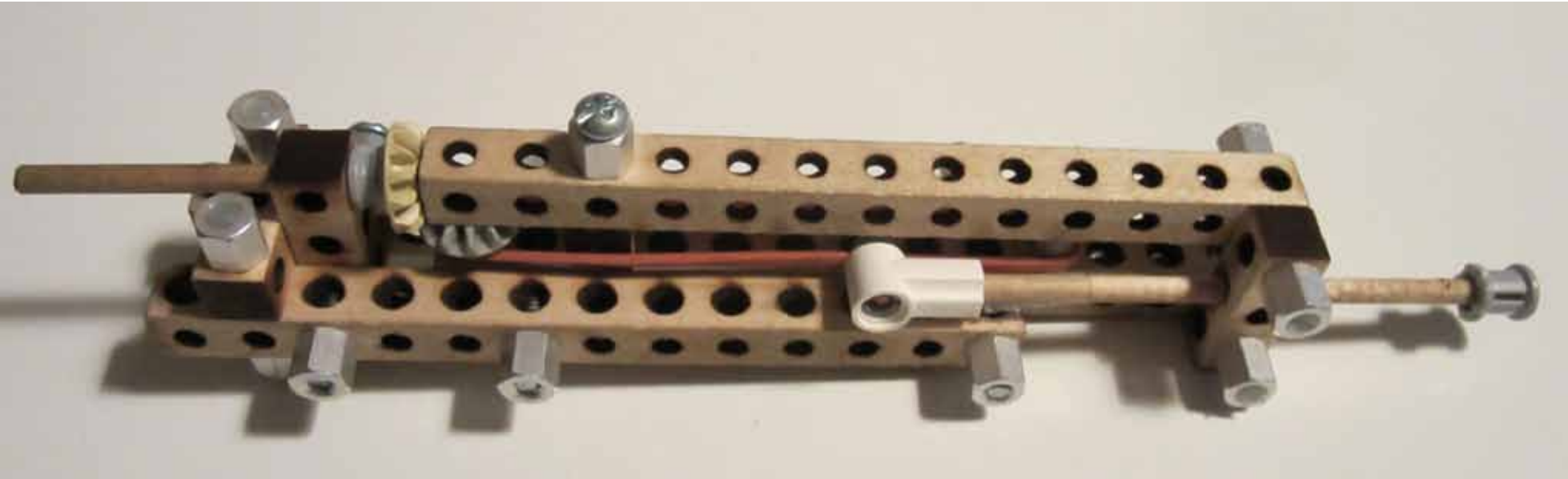
The other style is grey box testing. It's a



BitbeamBot the mobile testing robot

more precise approach, but it requires access to and internal knowledge of the application. I'll implement this approach by extending the Selenium library and tethering the phone to my computer via a USB cable. With the USB debugging interface turned on, I can ask the application precisely which elements are on screen, and where they are before moving the BitbeamBot pin to the right location.

I am documenting my building progress at bitbeam.org. The hardware, software, and mechanical designs are open source and available at <https://github.com/hugs/bitbeam>. □



The PinThing prototype

AUTHOR PROFILE - JASON HUGGINS

Jason Huggins co-founded Sauce Labs and currently leads product direction. Prior to Sauce Labs, Jason was a Test Engineer at Google where he supported the grid-scale Selenium Farm for testing Google applications such as Gmail and Google Docs. Jason's experience also includes time at ThoughtWorks in Chicago as a software developer. While at ThoughtWorks, Jason created the Selenium

testing framework out of the need to cross-browser test a new in-house time and expense system.

When not programming in Python or JavaScript, Jason enjoys hacking on Arduino-based open source hardware electronics projects. Although he was born in New York City, grew up in L.A., and travels in a gravitational orbit around San Francisco, Chicago is his kind of town.





Exploring scripted testing

By Sean Morley

The concept of Exploratory Testing, a term coined by Cem Kaner in 1983, has been an important development in the field of software testing. By approaching unscripted testing as a discipline with a certain mind-set, it can be studied, improved, and taught. (One of the benefits of applying a discipline to an otherwise nebulous subject is the ability for those naturally skilled to share those skills with the rest of us.)

I am relatively new to these concepts and am studying this area in an attempt to improve my skills. One topic I find interesting is the comparison of exploratory testing with its natural antithesis, scripted testing. I have seen comments that some testers have been so enamoured by the benefits of exploratory testing that they planned to abandon scripted testing altogether. This seems like an overreaction to me.

I would like to explore what I see as some of the main reasons for scripted testing, and the potential applicability of exploratory testing. Reasons for scripted testing:

- 1 A complex new application (or new feature within an existing application) where the complete functionality is not intuitive from the interface alone.

You need a thorough reading of the spec in order to document the procedures required to exercise the

new functionality. Exploring the tool interactively is not likely to achieve sufficient coverage.

- 2 A specialized application which requires domain-specific knowledge to fully exercise.

- 3 A tester who is new or simply unfamiliar with the application.

In each of these cases a scripted test, created by a knowledgeable tester, allows a less experienced tester to be effective - while gaining valuable experience with the tool and the subject matter. I find executing a test script and stepping through the usage scenarios a more effective way to learn a tool or feature than simply reading documentation. As the lightbulbs of cognition brighten, it will be natural to want to explore further and try your own examples - a good thing! Thus even the less experienced tester may use the test scripts as jumping off points for fertile excursions into extended test scenarios.

- 4 A new version of a mature product.

This script walks through the existing functionality in a reliable way to confirm that the new

development has not broken the legacy code. Note that this script is not testing the new functionality, but rather regression tests the functionality that existed in the last release.

Exploratory testing does not seem like an effective substitute for scripted testing when considering legacy functionality. The likelihood of finding as yet undiscovered defects in released code is lower, and care must be taken not to lose coverage offered by the scripts. Expounding on these scenarios into newly added functionality may be fruitful - but then that no longer fits into this category. However, regression testing like this could really benefit from automation.

- 5 Specific customer usage scenarios, especially where problems were reported in earlier versions.

This is a special case of the regression testing of legacy code discussed above, with the added gravity that they were supplied by or derived from a customer, possibly when a previous new release failed with serious consequences for them.

If the removal of one of these tests allowed a similar problem to slip by, you might as well stick an apple in your mouth, as the customer and your VP of Engineering will take turns roasting you on a spit.

- 6 The project schedule does not allow time to wait until "Dev complete" to begin testing.

Ideally, you work from the spec to create a test script which you will execute when the code is released for testing.

It's a natural tendency to want to see and explore the application as you read about a feature. And sometimes it can be a little difficult determining exactly how the software will react based only on a spec.

But the realities of compressed product cycles and tight schedules often make it necessary to overlap development and testing. A benefit is that you are more likely to get a clean, well formatted test script with good coverage of the documented requirements. Drawbacks include:

Continued on page 17

AUTHOR PROFILE - SEAN MORLEY

Sean Morley is a Principal Quality Engineer for Dassault Systemes. Having worked in a variety of roles on software tools for electronic design and test, he is currently focused on the testing of engineering software. Sean participates in test related forums and blogs at www.testyengineer.com in a concerted effort to improve his knowledge of the field and share his experiences. He appreciates all feedback, recommendations, and challenges as opportunities for growth, and can be contacted at seansync@gmail.com.



Continued from page 16

- The application may not function exactly as you interpreted from the spec.
- Design decisions may have been made during development that deviated from the spec.

The tester will need to work these out with the developer - and encourage a spec update.

Finally, when a test is developed from a spec you really should plan on a period of exploratory testing. No matter how detailed the spec, once you get your hands on the new feature you will likely see areas worth exploring further, which were not clear to you from a document with some screen shots and some text.



To facilitate outsourced testing.

As discussed earlier, testers unfamiliar with the product can be immediately productive and reasonably thorough, while also learning the application, if provided with a script. The next step in development of both internal and outsourced testers will be getting them writing their own test plans, helped by having already seen examples of good style and form in the test plans you supplied them. And again, this can serve as an excellent

SUBSCRIBE TO THE TESTING PLANET

FROM ONLY

£15 PER YEAR

<http://bit.ly/subscribehttp>

preamble to their own exploratory testing.



A test script that performs a walk-through of basic functionality of a new feature can be useful as an Acceptance Test.

This may be executed at various stages during the development cycle, later serving as the basis for a regression test.

Conclusion

Exploratory Testing is built on the premise that there is much to be gained by venturing out of the

sanctuary of pre-scripted tests. But can exploratory testing replace scripted testing? I believe that we have presented several cases where:

1. It is useful to have a test script.
2. It is beneficial to test following a script.
3. Those same benefits cannot be achieved with exploratory testing.

And yet in each case exploratory testing offers synergistic benefits. It seems clear to me, then, that neither technique can completely replace the other. In fact, they are complementary, and both deserve a place in your arsenal of software testing tools. □

How time is spent in software testing projects:

Updating a huge Excel sheet full of bug reports, until you feel like crying.



Actual bug tracking

Learn quick, test fast. Use ReQtest.
www.reqtest.com



Toll free US and Canada: 1866-608-9742 | Toll free UK: 0800-098-8452 | Other countries: +46 (0) 8 586 178 90



Some disruptive stuff: auto cheaper than manual, crowd/dogfood more effective than testers, rerecord faster than maintaining.
#gtac2011 @geomealer



Mobile testing – That’s just a smaller screen, right?

By Stephen Janaway

Introduction

There is no denying that mobile phones, smartphones in particular, are hot these days. Mobile phones have become an essential part of daily life, and as an essential part of life, they also need to be reliable, secure and easy to use. Imagine if the alarm on your phone didn’t go off in the morning, your old texts suddenly went missing, or you couldn’t make that important call.

As mobile phone usage has increased and mobile applications have become more and more important, so has the need to test them become more important. It’s an exciting part of the industry to be in. So if you have ever thought about a role as a mobile software tester, then read on and find out what it’s all about.

Mobile Software Testing Has Many Flavours

Mobile phones are pretty complicated things. Under the hood there is a lot going on, and so as a software tester it’s important that you decide early on where you want to focus your efforts. Broadly speaking you are looking at:

- Hardware level testing – this is testing at the device driver layer, where the software controls the interface to the processors, screen, memory and the radios themselves. This is a highly specialised area and somewhat complicated to get started in. Testing is primarily performed as part of development work, low-level, and automated.
- Protocol testing – your mobile phone contains a number of different radios in it that enable it to

talk to the mobile phone networks in different areas and countries, and also for Bluetooth, Wifi, etc. Testing at this level is sometimes called protocol stack testing, and means testing against various universally agreed specifications (an example would be 3GPP), using some pretty cool and pretty expensive pieces of equipment which simulate what the networks are doing. It can also be varied work, and can mean some fun getting out of the office doing drive testing. This is where you drive a specific route, where the network signal is known to change, and record and analyse the results, to see if the device you are testing is able to work in all sorts of different signal environments. And, based upon my experience, you also get to spend a lot of time eating out at service stations.

- Applications testing – this is probably the easiest area to get into if you have a background in software testing in another field. Mobile phone application usage has exploded in recent years, you can get an application for almost everything now, and we all expect those applications to work well. Applications can be embedded in the phones firmware, meaning they are available when you buy the phone, or downloaded from app stores. This is a great area of software testing to get involved in so let’s look further at this area.

Testing Applications

Firstly, what’s different about testing mobile applications when compared with working in the desktop world? On the surface not a lot. Your software testing skills will treat you well here. Remember those equivalence partitioning and

boundary value analysis techniques you learned on that ISTQB course you did? They’re just as useful here as if you were testing a desktop based application. But some techniques are different, or just require a bit more focus.

Probably the most important thing to take into account when testing a mobile is that the software that you test could be in the hands of millions of people once you have finished. Releasing a phone with bugs in it, which mean customers need to return the phone, loses a company a huge amount of money. The mobile world is a low margin, high volume game. There are few players and brand is important. The mobile applications world is also very competitive, and customers can give feedback straight away, into the applications stores, for all to see. Good quality is extremely important.

For mobile applications a lot of testing is performed using emulators or remote device access services. Remote device access services enable you to connect over the internet to real devices which are provided by other companies for your testing use. However, final testing should always be performed on the devices that the customer will either buy in the shops, or download the application onto. There’s too much going on below the application level to just use the emulator. So even if you use emulators a lot, you should still get to play with the cool kit sometimes.

Interactions Are Important

When mobile phones started they were pretty simple. Calls, texts, and maybe a few games and not much else were about all your phone could do. These sorts of phones have not gone away, but even these are more complicated than they look, and smartphones are even more so.

Look under that UI, and under that client, there’s a whole software stack running. While your application runs, the phone is still talking to the network, working out where you are, maybe receiving calls, texts, Facebook and Twitter updates. There are a lot of interactions going on that you don’t see.

Understanding how the OS of the device you are testing (iOS, Android, Symbian, Windows Phone 7, etc.) works is important. You don’t need to be an expert but you do need to be able to design tests that focus on these interactions. What happens when you have your application running, and then get a call, and a text, whilst uploading video to YouTube? Test for it and find out.

Performance Is Limited

Mobile devices have limited memory and limited processing capacity, and so it’s important to test how the devices work when stressed. Fill up the phonebook, the photo gallery, etc. Then see how well the device performs under pressure. If you are testing an embedded application, maybe a ‘feature phone’ rather than a smartphone, then test for memory leaks and general performance of the operating system and applications. There’s not

Continued on page 19



much memory on a cheap phone and the processors are very slow.

It's also important to think about battery life. How power hungry is your application, does the battery run down quicker with it running, what happens to it when the battery runs out? Good battery life is very important to users, we've all been in the situation where you need to make that important call but the phone is dead.

Third Party Compliance

You may need to test for certification compliance; all phones are certified to certain standards. Without meeting these standards then it's not possible to sell a particular device, or have a particular application accepted into an apps store. These include:

- 3rd party software runtimes – most, if not all devices are capable of running Java Mobile Edition applications and some others support runtimes such as QUALCOMM's BREW platform. This requires that certification test sets are run and all tests pass.
- Network operators have certain technical acceptance test sets to pass before they will start selling devices. Since most devices are sold by operators then passing these test sets is very important.
- Most mobile applications stores have certain submission criteria; does the application meet these?

Usability Matters

You should test for usability. Usability means testing it on a panel of real users. First of course test it yourself, but then find a group of typical users and observe them interacting with the device or application. Typical areas to focus upon are:

- How easy is it to use the application? Do users

AUTHOR PROFILE - STEPHEN JANAWAY

Stephen Janaway has been working in software testing for over 12 years, focusing on the mobile device and applications area. He's worked for companies such as Ericsson, Motorola testing and test ma is currently heading quality assurance g focusing on mass m



Stephen blogs on his favourite subjects (software testing and Agile techniques, particularly Kanban) at www.onegreenhill.blogspot.com and you can follow him on twitter @onegreenhill.

‘just get it’ when they first start the application or pick the device up or do they need training? Difficult-to-use applications are often downloaded from app stores and deleted almost as quickly once the user finds they actually need to learn how to use them.


- How well does the application fit the small screen? Is the text easy to read and is that button in the right place? Look and feel, especially on a small screen is very important.
- Does the application or device feel fast? Do the users feel they are waiting too long for a screen to load or response to be received?

Usability is more important in the mobile world; the screen is smaller, it's more difficult to interact with it and mobile devices are typically used one handed. The difference between a device or application which has good usability and one that does not is very obvious. It's what's led to the huge success of the iPhone which has excellent usability.

Automation Is More Complicated Than In the Desktop World

Automation is also a very important testing discipline to consider. Using automation for regression testing for mobile applications and devices is a very cost-effective strategy. A mobile application or device regression testing suite can be huge, with many interactions between other applications and areas to consider.

Automation in the mobile world is not as easy as it is in the desktop world. Your device is, after all, not able to run a large automation tool on its own. There's not the memory to install it. The UI of typical mobile devices is highly dynamic; the user can use swipes and other gestures on the screen to interact with the device. This means that automation solutions typically run using emulators, or are bespoke systems running on the devices themselves. But the good news is that there are more mobile specific automation tools beginning to become available. A lot of automation testing for mobile phones takes place below the UI level as well, making it easier to drive the tool from the desktop, and then communicate with the device via



Find Cost Savings Without Sacrificing Outcome

Edited by
Matthew Heusser and Govind Kulkarni


Plenty of software testing books tell you how to test well; this one tells you how to do it while decreasing your testing budget. A series of essays written by some of the leading minds in software testing, **How to Reduce the Cost of Software Testing** unveils powerful tips, tactics, and techniques to help you accelerate the testing process and improve the performance of your test teams, while lowering costs.

- Considers the latest advances in test automation, ideology, and technology
- Describes the economics of starting a test team within an existing development group

SAVE 25%

and receive **FREE** shipping when you order online
and enter promo code **KVK04**

Catalog No. K12845
September 2011, 340 pp.
ISBN: 978-1-4398-6155-4
\$69.95 / £44.99



CRC Press
 Taylor & Francis Group

www.crcpress.com

USB or Bluetooth.

So Should I Get Involved?

So, is mobile phone testing just like testing on the desktop but a smaller screen? Do you just need your existing skills and maybe some glasses? Yes and no. Techniques you already know are equally useful in the mobile world. But how you apply them, and what you do in addition to them, is more important.

Working in the mobile applications and mobile phone testing world can be really varied work. If you work for a manufacturer or network operator then you get to test using a lot of new, prototype kit, especially when testing the devices themselves. Maybe you always get to have the latest phone. The work is fast paced, and you can quickly see the results of what you do, either on the high street or in the apps store.

As someone once said to me “Be proud, your work will soon be in the hands of millions”. It’s very satisfying to see someone in the street using a phone that you worked on, and knowing you played a part in making sure the software was high quality. It’s the quality of the software being released that decides if those millions love their new phone, love their new application, or if they return the phone or delete the application. No pressure on us testers, right? □

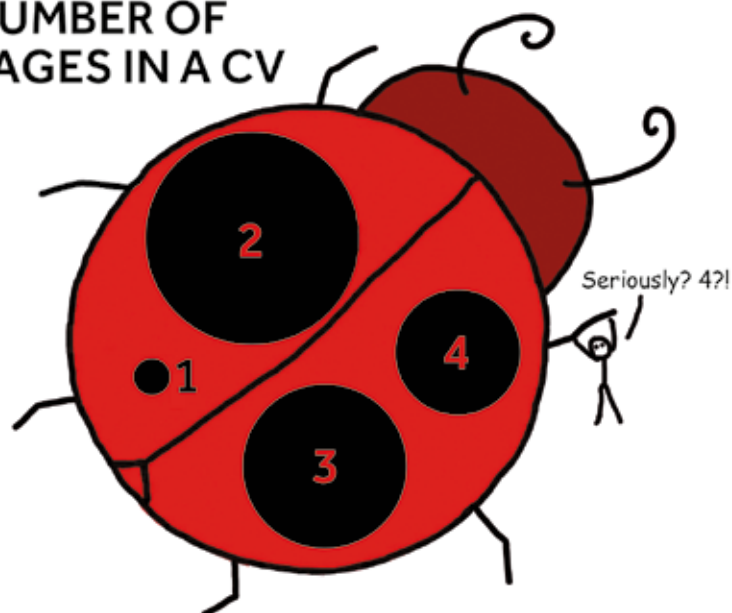


"Stop thinking of the page as a cohesive whole." #gtac2011 @marlenac

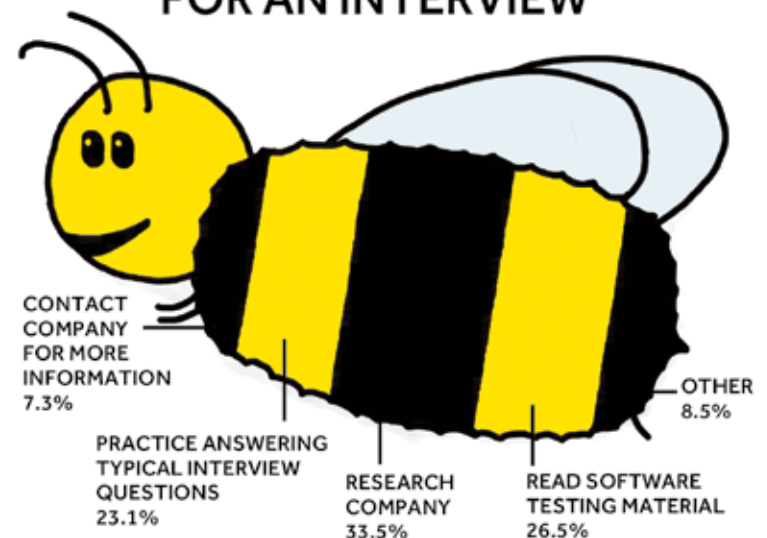
APPLICANT &

A RECRUITMENT SURVEY ON APPLICATIONS

NUMBER OF PAGES IN A CV

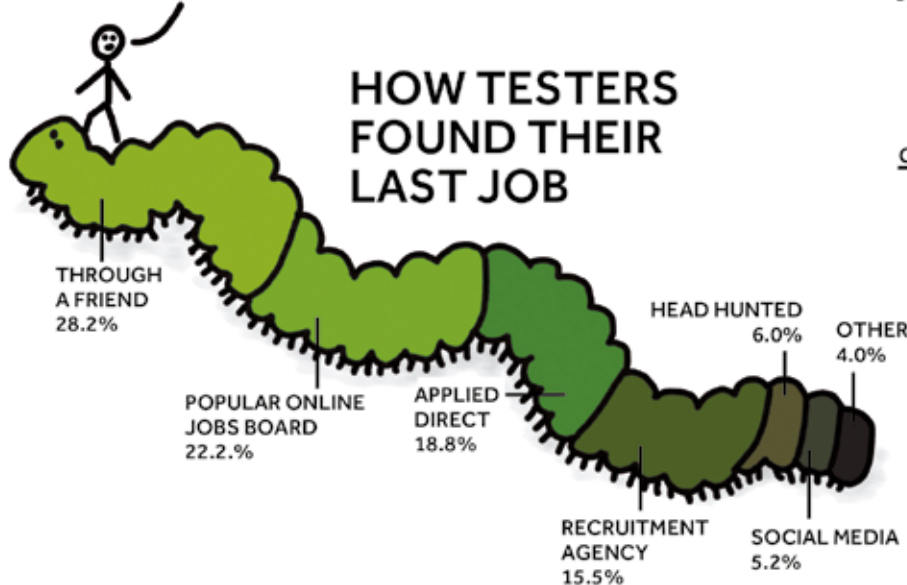


HOW TESTERS PREPARE FOR AN INTERVIEW

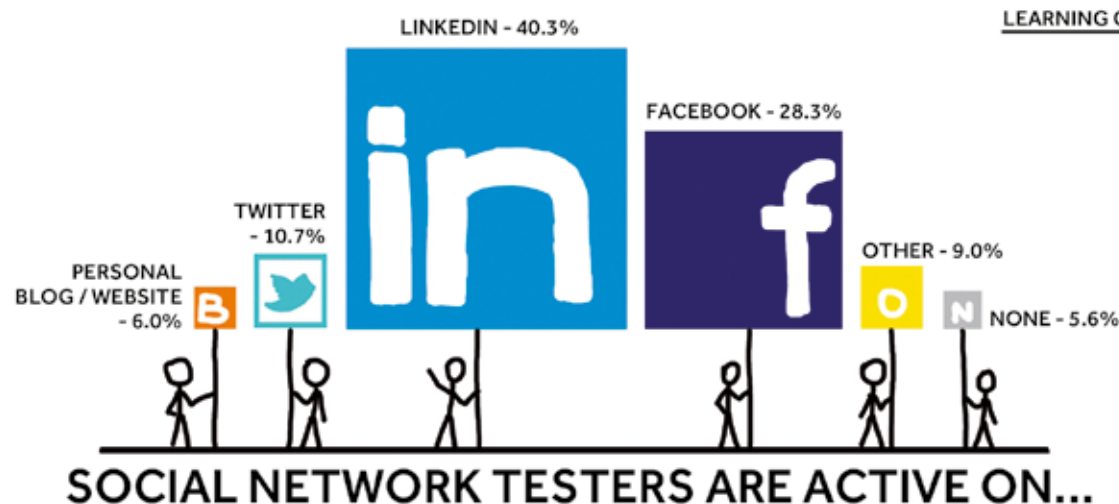
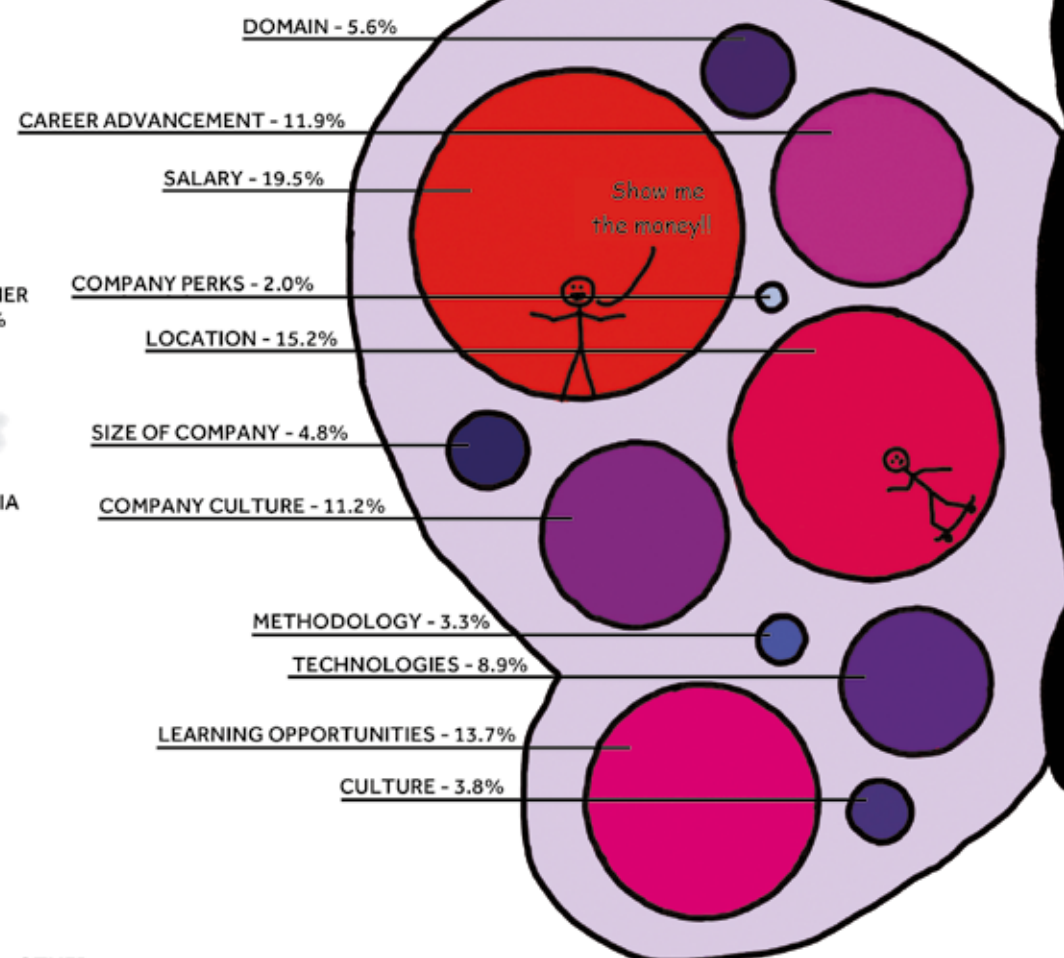


Mate, it's not what you know, it's who you know!

HOW TESTERS FOUND THEIR LAST JOB



IMPORTANT FACTORS FOR TESTERS WHEN APPLYING



Sponsored by...



Software Testing Europe.com

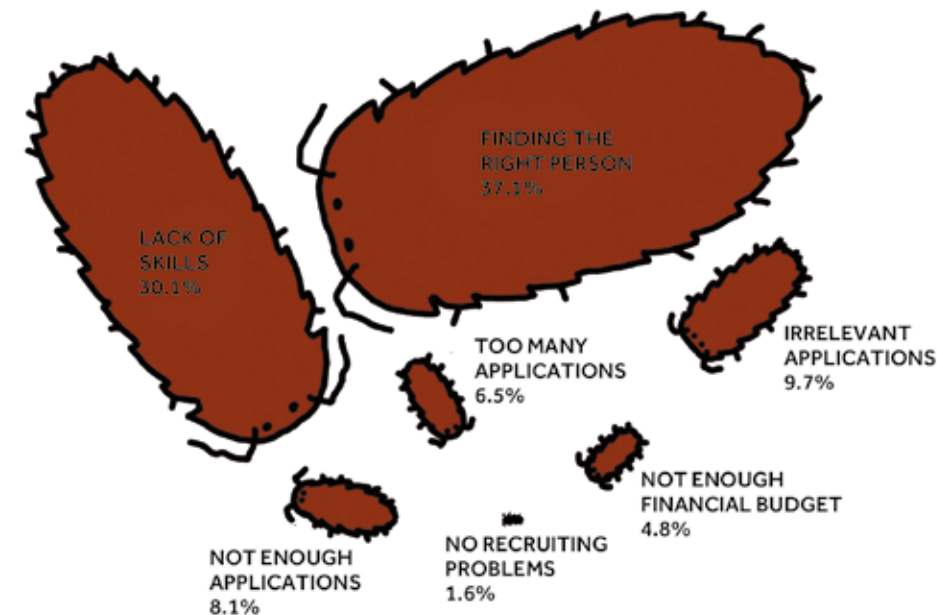
& EMPLOYER

AND INTERVIEWS FOR SOFTWARE TESTERS

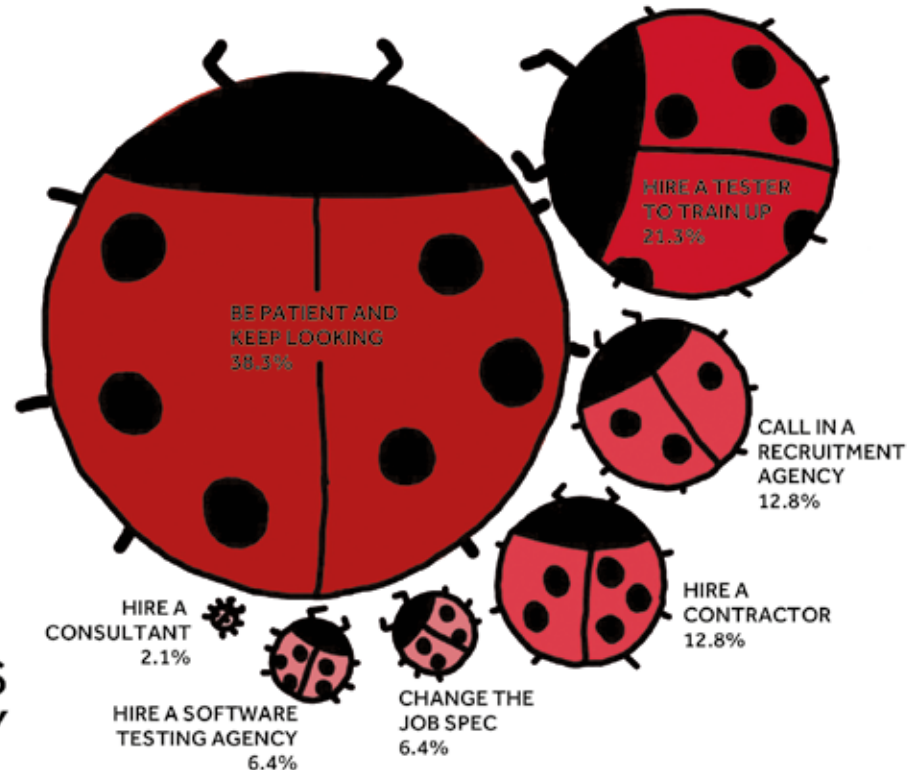
I'm not asking for much...
Just a tester to work
better and quicker than the
last one... and for cheaper!



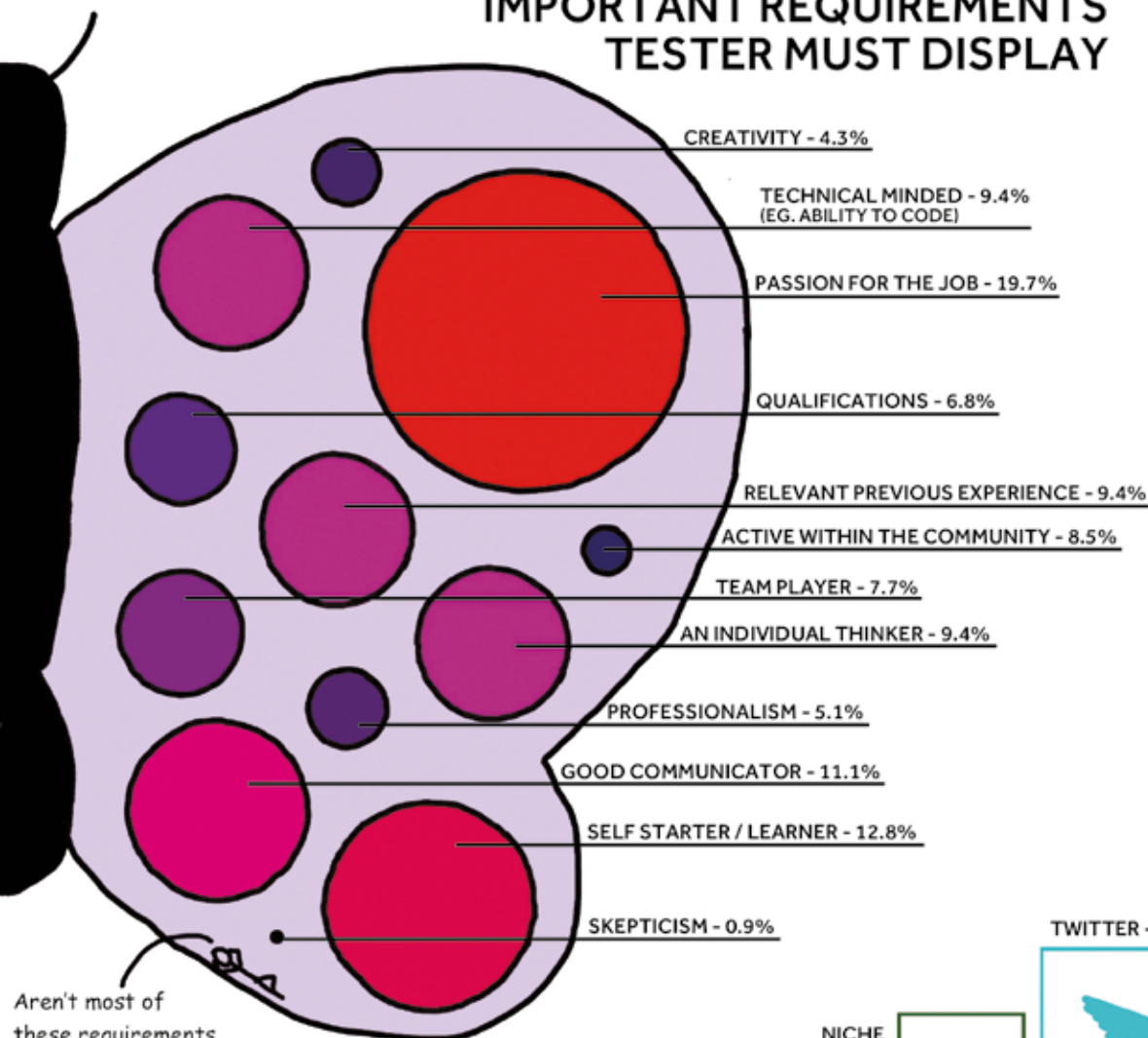
BIGGEST PROBLEMS IN RECRUITMENT



PLAN B WHEN THE RIGHT TESTER CAN'T BE FOUND

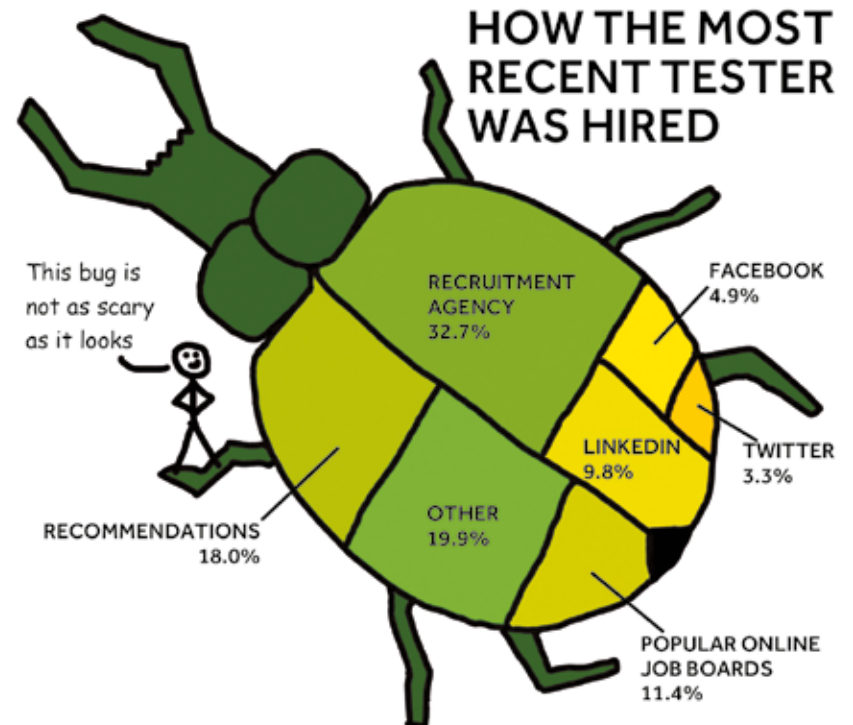


IMPORTANT REQUIREMENTS TESTER MUST DISPLAY



Aren't most of these requirements good for any role, not just testing?

HOW THE MOST RECENT TESTER WAS HIRED



This bug is not as scary as it looks

LINKEDIN - 31.9%

TWITTER - 21.3%

FACEBOOK - 21.3%

NICHE SOFTWARE TESTING COMMUNITIES - 14.9%

OTHER - 13.4%

Testing & Communication

By Lisa Crispin

I just read a Twitter exchange between @michaelbolton and @adampknight where they mentioned that “testing” problems are often really communication problems. Personally, I find that a big part of my job as a tester is facilitating conversations between the development team and business experts. Different people may have different interpretations of the same set of requirements, but don’t realize they’re not all on the same page. When something like that happens, I try to get all interested parties together – preferably with a whiteboard – to talk about it.

Here’s a recent example. We had just started work on a new theme, and one of the user stories had particularly complicated and confusing requirements. A programmer came to tell me that he and another programmer felt that the story was unnecessary, as it was designed to capture information that wasn’t used anywhere else in the system. He said the story’s main stakeholder had agreed, though he hadn’t been able to talk to our product owner yet. I had a feeling that the stakeholder was forgetting something.

Our product owner is a busy person, but

I finally tracked him down later that day. He still believed the story should be done, with the original requirements. I went and got the stakeholder and the two programmers, and we congregated in the product owner’s office to talk about it. After a ten minute discussion, everyone understood that the user story was still needed, but the most complex part of the requirements could be dropped, and a much simpler solution implemented. This enabled us to finish the story in about one-third the time, while still satisfying the business needs.

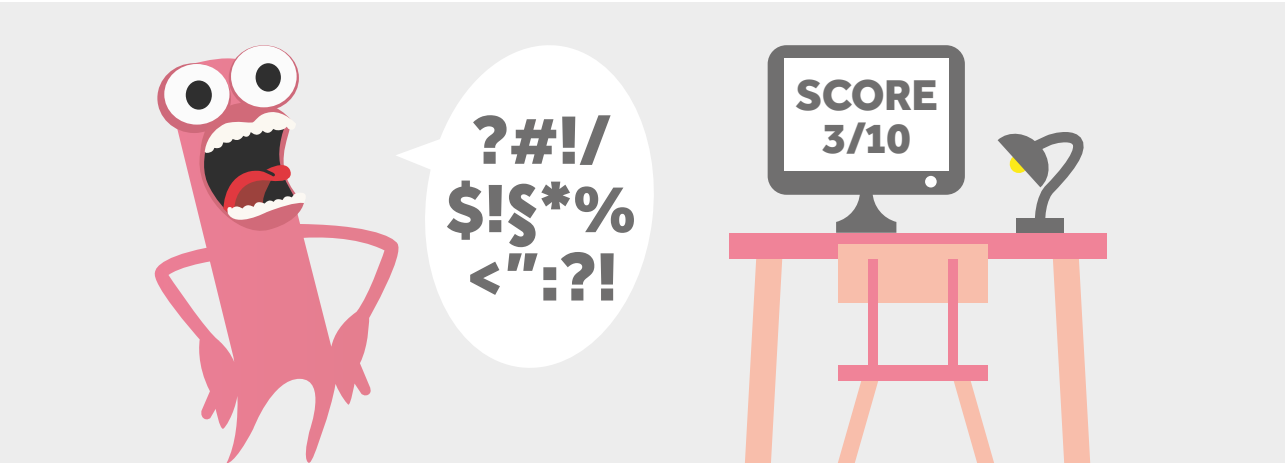
Sometimes when I tell a story like this, people ask, “Is it really your job to call impromptu meetings? Shouldn’t the manager or Scrum Master be in charge of that?” In my opinion, when you follow the whole-team approach to software quality, anyone should feel free to get all the right players together for a quick conversation whenever needed. Testers are always looking at the software from multiple points of view, thinking about the value each user story brings to the business, how end users will behave, and whether the technical implementation is appropriate. So, we often catch missed assumptions or incompatible requirements.

If I suspect miscommunication, I don’t start an email thread. I get up and go talk to people in

all the different roles that need to have input. We gather together, draw on the whiteboard, video chat on Skype with any interested parties who aren’t physically in the office, and resolve the issues. Good communication early and often saves a lot of time and bug reports later! □

AUTHOR PROFILE - LISA CRISPIN

Lisa Crispin is the co-author, with Janet Gregory, of *Agile Testing: A Practical Guide for Testers and Agile Teams* (Addison-Wesley, 2009), co-author with Tip House of *Extreme Testing* (Addison-Wesley, 2002) and a contributor to *Beautiful Testing* (O’Reilly, 2009). She has worked as a tester on agile teams for the past ten years, and enjoys sharing her experiences via writing, presenting, teaching and participating in agile testing communities around the world. Lisa was named one of the 13 Women of Influence in testing by *Software Test & Performance* magazine. For more about Lisa’s work, visit www.lisacrispin.com.



Testing the tester

A short story about practical recruiting methods for testers

By Mitch Goldman

The Setup

“I’ve got good news and bad news.” chirped Amelia, our HR director, “The good news is that people are responding to your job posting for a new software tester.”
“That’s what we want, isn’t it?” As a test manager for many years, I’d hired plenty of testers

over the years, and I knew the drill. “We post the job, and testers apply for it.”
“Sure, but the bad news is we got a dozen so far and it’s only been the first day, and that doesn’t include what we’ll get from recruiters. You’ll have a lot of candidates to sift through. What’s your plan?”
Okay, new drill. In the past I’d only gotten a handful of applicants, so it was easy to do phone screenings for all of them. With this many applications,

I was going to have to think of something new. “No problem,” I said, “I’ll get the team together.”

The Big Idea

“So that’s the problem. Most of the candidates meet the job description, and we can’t interview all of them without taking up too much of our time. We’re going to need another way to evaluate them. Any suggestions?”
“Only pick the ones with years of experience in our field!” a developer said. “Or years of experience on our test tools,” a tester added.
“If that were the case,” I said, “most of you wouldn’t have been hired. None of you had the exact experience when you started, but that didn’t stop you from learning and doing a good job.”
“Puzzles!” someone shouted out. “Only interview those who can figure it out.”
That raised a few grumbles, “Ugh, no. I never liked interview puzzles,” said the senior developer, “They always seem like trick questions, and they usually have no relation to how someone does real work.”
“Good point,” I said. “But I like where you’re headed. So how can we find out how they’ll do real work?”
“We’re testers!” said a senior tester, “Let’s do what we do best... design tests for them!”
“Testing the testers. I like the sound of that. All in favor?”

Continued on page 23

Continued from page 22

The Test Planning Test

“Okay, where should we start?” I asked.
“At the beginning,” said the senior tester.
“The first thing that we have to do on a project is to write a test plan. We should be sure that they can write one, too.”

Candidate Test 1. Write a test plan outline for a real project.

Write a brief summary of one of your recent projects. If the project was very large, pick a smaller slice of it to summarize. Then ask the candidate to write a test plan outline, no more than a page long, of how they would approach the testing of that project. *Did it cover the same areas that you had to cover? Did they think of new areas that should have been covered? Was it an outline that you would show your Test Manager or Project Manager?*

The Test Design Test

A developer raised an objection. “A test plan is a good start, but that still doesn’t show if they’re any good at designing tests to cover the stories.”
“Perhaps,” I said, “So what test do you propose to find that out?”

Candidate Test 2. Write acceptance tests for a user story.

Give the candidate a user story from one of your recent projects, and ask them to design the acceptance tests for it. *Did they design them like you did? Would the tests have covered the story enough so that the resulting application would pass acceptance tests by the customer or end-user? Would you show the acceptance tests to your customer/end-user?*

The Non-Technical Communication Test

“What about soft skills?” the project manager chipped in. “For instance, what if that user story didn’t have enough information to write the acceptance tests? They have to talk to the product owner or end-user to get the information, and they’re usually not very tech-savvy.”

Candidate Test 3. Write an email to the customer/end-user asking questions.

Using the same project summary and user story in the previous Tests, ask the candidate to write an email to a non-technical person and ask questions that would have helped them write a better test plan and acceptance tests. *Was the letter clear and did it ask good questions? Would you have been satisfied to send it to the customer/end-user, with minimal editing?* [Note: this test can easily be modified depending on your context. You may want your candidates to write an email to a more technical person, such as a specialist software engineer or the project’s architect, instead of a less technical one. Or maybe go all the way to the top, and ask them

aim the email to the CEO of the company as if they were the end-user.]

The Exploratory Testing & Defect Writing Test

“We’re beating around the bush,” a junior tester spoke up, “Most of their day will be spent testing and finding bugs. That’s what we should find out if they can do.”
“Great point!” I said, “What did you have in mind?”

Candidate Test 4. Do an exploratory testing session. Write up the testing notes and the best bug found.

Provide a real application for the candidate to test. (For ideas, check out <http://weekendtesting.com>.) Ask that they do an Exploratory Testing session, giving them a mission and a fixed duration, 20-30 minutes should do. At the end they must provide the session notes for a debriefing, and a bug report of the best defect that they found. *Were the session notes clear? Did they follow the mission and test it well? If they strayed, did they have a good reason? Was the defect written clearly and with enough information? Was it a good defect, the “best bug” that they could have found, or a superficial one?*

The Automation Coding Test

“We can’t ignore our automation test tools,” said the senior automation tester. “They all say that they’ve got experience, but we should design a test to identify the difference between beginners and advanced users.”

Candidate Test 5. Write an automated test using a specified test tool, or pseudo-code if not available.

Give the tester a real-world scenario and ask them to write an automated test for it. The scenario shouldn’t be too complicated or time-consuming, but also not so simple that any beginner could do it quickly. If your tool isn’t available for candidates to download and try for free, you could ask them to write it in pseudo-code. *Did the automated test run without errors? Did it meet the objective of the test? Was the code logical and well written? Did it have comments and error handling?*

The Paired Testing Test

“I’ve got good news and bad news,” chirped Amelia from HR. It was one week later, and she called another team meeting to update us on the recruiting. “The good news is that only 30% of the applicants were serious enough to complete the tests. You’ve been evaluating their results, and the list has been narrowed to three candidates.”
“So what’s the bad news?” asked the project manager.
“We’re out of tests. Is there any other evaluation that you want to do?”
The room was quiet for a moment, then I had an idea. “When we’ve finished a user story,

AUTHOR NOTES

This article contains only a few ideas on how to “test the Tester.” What you choose to use should depend on the context of your company and the role that the tester will play. I’d expect different tests for recruiting a junior tester at an insurance company than I would for a senior automation tester for mobile apps.

In my experience, the recruiting process is “top-down”, starting with general details and drilling down to specific ones. It starts with CV reviews, then a phone-screening, then an interview and maybe, lastly, some actual hands-on tests. I prefer to turn that process around and do it “bottom-up.” Give all the candidates some hands-on tests to perform, even those whose CVs might not seem to be a good match. Focus on the skills that they’ll actually be using day to day, and the quality of their work. Then move up to meeting them and doing the CV check. The results may surprise you, and the team you build should be stronger for the diversity.

I also find that using a process like this yields candidates who are much more invested and eager to join the company because they’ve gotten a feel for the work they’ll be doing and the challenges they’ll be facing. The response rate to your tests will probably be low, but if they’re not up to the challenge of doing the tests, they’re probably not someone that you’d want to hire. For the ones that do respond, it will be much easier to spot the top performers and allow the best candidates to rise to the top.

My inspiration for this article:

- Weekend Testing - www.weekendtesting.com
- PairWith.Us - www.pairwith.us

These groups meet in their spare time to collaborate on practicing the craft of testing, for learning and fun. They inspired me to change my recruiting practices, because I learned that working together is the best method of evaluation and therefore should be at the forefront of the recruiting process, not the end.

- I also recommend Cem Kaner’s detailed paper “RECRUITING SOFTWARE TESTERS.” www.kaner.com/pdfs/JobsRev6.pdf Cem does his usual thorough job of academically covering more areas on this subject than I could ever hope to know.
- I do not recommend the process described in the Recruiting section of The-Software-Tester.com. www.the-software-tester.com/Jobs/software_tester_assessing_job_applications.html They propose creating an essay-based “Pre-Interview Exam” to test the tester, but I don’t find that to be a very effective way of identifying top performers. I provide this link as a counter-point to my opinions.

Continued on page 24



AUTHOR PROFILE - MITCH GOLDMAN

Mitch is a project manager and the test manager at IBuildings, where he hires all their testers using the methods in this article. He has been managing QA test teams and delivery for over 15 years. He is focused on Agile development methods and finding ways to bridge the gap between customers, developers, and testers to ensure that projects are delivered successfully. Mitch has featured as a speaker at the StarEast conference and contributed to several London Exploratory Workshops in Testing (LEWT). His current pet projects are automating Magento e-commerce stores and learning how to use Selenium Webdriver using PHP.

Continued from page 23

‘done,’ what do we do next?”
“Show it to the product owner for User Acceptance Testing,” replied the junior tester.
“Right.” I said. “We know that the remaining three candidates are qualified. What we want to know now is if they’re someone we’d want to work with. So let’s work with them and find out.”

Test 6. Do some paired testing.

Call in the short list of candidates and have them pair up with someone on the team to do a specific task, such as testing a feature, writing acceptance tests, or automating a script. Work side-by-side with the candidate for a timed session, then debrief it with them. *Did they accomplish the task? What was the quality of their work? How did the pair get along? What were the major obstacles, and how did they overcome them?* [Note: this can easily be combined with the exploratory

testing or automation coding tests. If you have the time, I would strongly recommend this approach because you will be able to evaluate candidates much better if you’re paired with them during these exercises.]

Conclusion
“I’ve got good news,” chirped Ameila, “The candidate accepted our offer. They had a number of offers on the table, but they chose to go with us.”
“That’s great!” I said, “Did they say why?”
“Our interview process. I thought it would be too difficult. But of all the places they interviewed, they said that we were the only ones who seriously challenged them. Also, because of all the real-world scenarios we gave them to work on, they said they knew exactly what they’d be signing up for when they joined us. That’s what gave us the edge.”
I hesitated to ask, “So what’s the bad news?”
“None,” she smiled, “Why do you always think I’ve got bad news?” □

Mapping testing...

By Albert Gareev

I am a follower and strong proponent of the exploratory, context and heuristic driven testing approach. From here, I could go on for quite a few pages telling you how much I found it powerful, flexible, enjoyable, challenging, educational, and so forth, but instead I’m willing to share a story of a very special challenge about exploratory testing: the challenge introducing it to someone who is only used to a scripted approach, and the challenge of documenting a dynamic, non-linear exploratory test.
This story is based on my real experience; however, all context specifics have been removed or replaced with fictional content.
How would you teach someone to ride a bicycle, someone, who has never tried it before, without giving them a bicycle to practice? Sure, you can tell them about what moves to make, even tell them the theory of static and dynamic balance, you could even give a ride to that person, but it still won’t help build their skill of riding a bicycle.
Demonstrating exploratory testing is no different from demonstrating a ride on a bicycle. People can observe an external part: mouse moves, keystrokes being typed, navigation actions, - seeming to be quite random, ad-hoc, “without a purpose”. They can’t see what’s happening in your mind, can’t see test ideas sparking in your head, and models being structured and populated while you’re learning about the application. Written notes may partially help, and it’s helpful to comment your actions while you do. And yet creating a ‘user guide’ is nearly impossible, as it’ll be about fixed assumptions and pre-scripted steps.
What I have found is that the structure of a mind-map can help in capturing a ‘slice’ of

the dynamic exploratory testing process: it allows keeping a non-linear structure, it’s rich, and it can always be expanded further. The story begins with a ‘simple’ test case...
One morning I asked Marge, a tester in my team, were there any problems with address form in the application, with the postal code in particular. “No”, - she said. - “There were two test cases for the postal code, both passed”. Intrigued, I asked what test cases those were, and did she think of some other tests? “There was one for valid and one for invalid, and, I think, that’s it”, - Marge replied. “What if the application rejects some invalid characters and misses others?”, - I asked. I then started coming up with some more test ideas off top of my head:

- Does it accept both upper- and lower-case?
- Must space character always be used as a separator, and are there any other allowed separator characters?
- Does the application verify if it’s an existing postal code?
- Does the application validate postal code against the other address fields entered?

Marge looked confused, so I came to her aid. “These are questions about the actual behaviour of the application. Most of these questions are not reflected in the requirements to an extent of explicit ‘expected result’, but we can find it out through testing. Without requirements, we might be unsure whether the behaviour is ‘right’ or ‘wrong’, but, to begin with, we need to find out what that behaviour is. Based on our findings, we may assess what might be impacted by a particular behaviour and how. This provides us with information to be able to

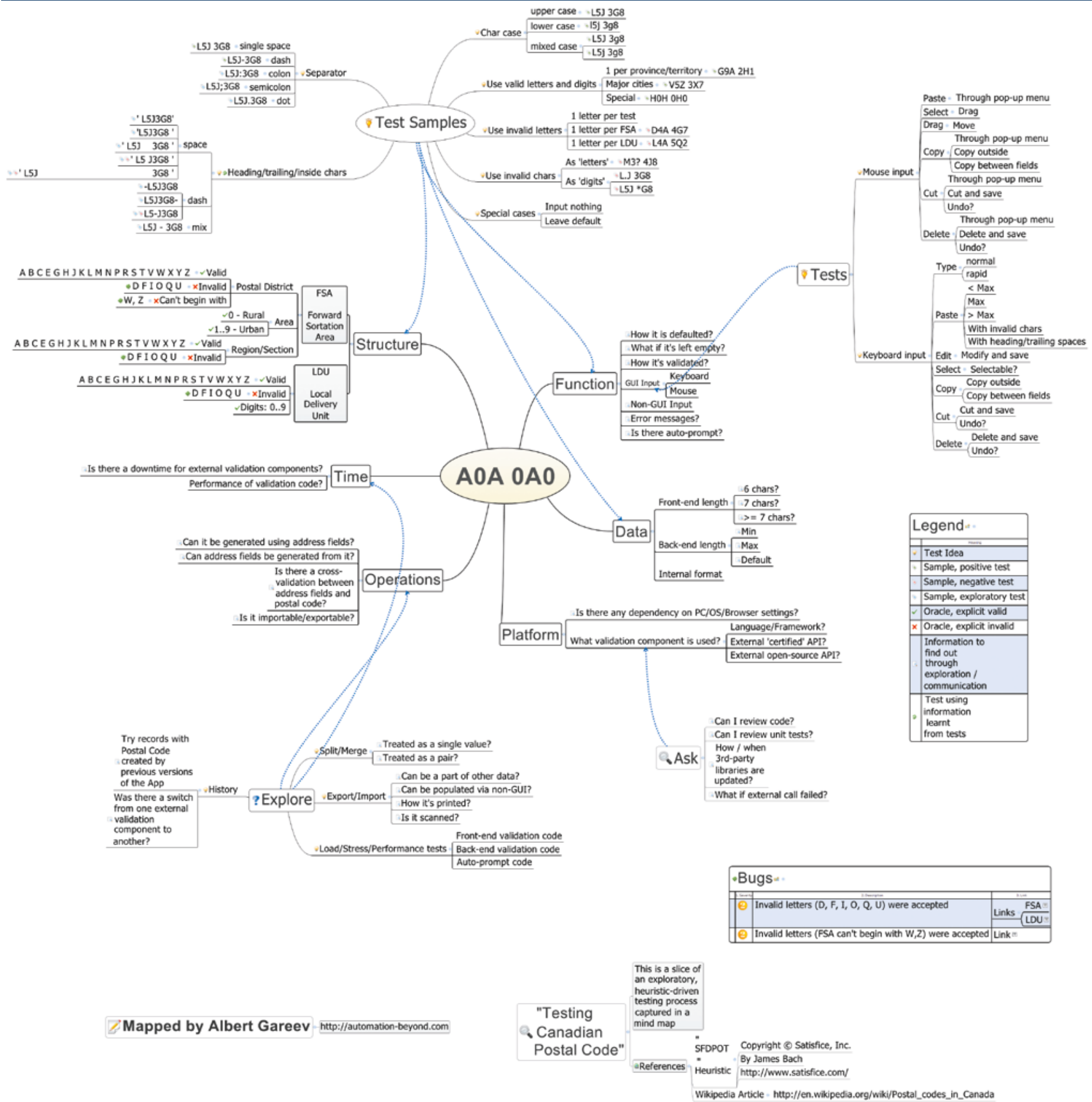
come up with next series of tests.”
“First, let’s write down what we know and what we want to know about postal code, in general, and with regards to our application. To guide our investigation, we’ll use Structure, Function, Data, Platform, Operation, and Time categories. These heading make ‘SFD POT - San Francisco Depot’, a mnemonic and testing heuristic created by James Bach.” (<http://www.satisfice.com/articles/sfdpo.shtml>)

Structure
What is a Canadian postal code, really?
Google search is a very handy ‘testing tool’ for such investigations, and the first match is an article from Wikipedia (http://en.wikipedia.org/wiki/Postal_codes_in_Canada). There it is - we’ve learnt about FSA (Forward Sortation Area) and LDU (Local Delivery Unit). We soon discover that not all letters can be used - “D, F, I, O, Q, and U” are invalid!
Going back to the application, a quick test - and voilà - the postal code with invalid letters is accepted as valid! Let’s put down a note about this bug, and let’s make a note to try using all invalid letters, one per test, and move on. Actually, “W and Z” can’t be used as a beginning character, so a couple of more tests to try, plus we’ve got plenty of valid combinations to use, especially with the link to an online postal code look up and validator tool, provided by Canada Post (<http://www.canadapost.ca/cpotools/apps/fpc/personal/findByCity>).
Another question to ask is do we have similar validation functionality in our app?

Continued on page 25



MAPPING TESTING...



Continued from page 24

Functions

What functions are connected to the Postal Code field in the GUI, and at the back-end? Checking the source of the web page reveals that there is some front-end validation JavaScript, which aims for invalid format. Note: we can unit test it,

and we can ask programmers if they unit test the validation code. Validation may trigger an 'error message' response, so let's put a note to find out if the application is supposed to recognize different kinds of invalid postal code, and what and how it reports for those cases. On the go, we mark 'default' and 'reset' micro-functionalities for the further exploration. Let's not forget to put a note to investigate further how the application suggests

different default postal codes.

Certain time we spend trying the input functionalities – type, paste, select, cut, copy, using both keyboard and mouse for input.

Some of our time we spend trying the input functionalities – type, paste, select, cut, copy, using both keyboard and mouse for input.

Continued on page 26

Continued from page 25

Data

We’ve learnt a lot already about the data we use to represent postal code: it’s a single line, it has a certain structure, but it might be left empty as well. What we’d like to know is how the data field is defined at the back-end, in database’s table (or tables), and how it is handled by the application in memory. What type? Length? As a single value? As a pair? Can it be a part of another field?

By the way, we need to learn more about the length. Is it six characters? Seven with a separator or even more? It turns out to be seven and more. What do we do with that? Let’s find out what can be a separator. Is there anything else but a space character? Okay, dash is accepted. But only single dash. However, space-dash-space combination is still accepted. At this point, seems like we learnt something new about the functionality, so let’s go back to see how all these different combinations with separator characters are stored and retrieved.

Functions

As we’ve figured with a few tests, the application is ‘smart’ enough to get rid of some meaningless characters, used as separators, or trailing/heading parts of the postal code. It also changes data to uppercase, regardless of input. Let’s see now, what purpose all these functions serve.

All these tests with data manipulation raise questions about different ways to receive the postal code. Is it only a GUI input? Can it be imported? (and exported as well). Through a file (What about format of data in the file)? Through a message (is there validation of data record)? Scanned, maybe? Overall, in which operations it’s used?

Operations

- Is it used as a reference data only or used actually in some automatic mailing procedures?

- Is it used to validate other fields or other fields used to validate postal code?
- Is it collected for some statistical analysis?
- Is it automatically or manually exportable to other applications?

Note, that answers to the questions above we need to obtain through communication with the project stakeholders. Once we know more details, we can decide on what we want to explore further. Some of our findings will reveal what operations and functionalities depend upon.

Platform

Certain ways the data displayed and operated are dependent on operating system’s settings, or configuration of the application itself. Let’s put a note that we need to find out about those dependencies.

From a quick chat with the programmer we knew that the front-end validation JavaScript was taken from some open-source example, and the back-end validation code consists of some in-house built functions and functions that are part of the framework. Make a note: that we must find out how carefully the code unit tested, do programmers run automated checking routines, and on what occasions they switch the validation code to new versions or another libraries. While it’s out of our control, being aware of that will help us to plan our testing effort.

By the way, how much the application relies on those external libraries? What will it do if some libraries were not installed, or an older version of the framework was installed? What if the external API call is failed? Is it considered as a passed or failed validation? Lots of questions to ask; let’s make a note of them for future reference.

Time

Although we’re exploring only a bit of the product, there might be timing issues

associated with it.
First of all, we’d like to find out was there a change in format, data validation rules, and were there any backward compatibility issues.

Second, we have a question (in fact, many questions) about performance of those validation calls, especially external, in various conditions of connectivity and under load.

Over an hour is passed. We have learnt a lot about the application, we have found a few bugs, and we’ve raised even more questions. Marge looks overwhelmed: “So much to test! And it’s only a single postal code field. And I have no idea how to test, for example, for performance issues”.

Yes, that is true. We can’t even think of all possible tests, and we for sure don’t have time for all of them to try. We may not have skills for some tests, and we don’t have facilities for other. But as long as we are aware of them we can and should make our project stakeholders aware, especially aware of those tests that have a feasible chance of revealing major threats to value of the product.

And to tell our testing story, the story of what questions were asked and answered, what was explored and found, we can use a mind map. □

AUTHOR PROFILE - ALBERT GAREEV

Albert Gareev is a dedicated testing and test automation professional, technical lead and a hands-on consultant. He combines passion for testing as investigation and learning with an automation approach helping to reach observation levels beyond “naked eye and hand” in complex, large-scale systems. Albert has a blog (<http://automation-beyond.com/>) dedicated to his professional passion and occasionally tweets at @AGareev.

SUBSCRIBE TO
THE TESTING PLANET
FROM ONLY
£15 PER YEAR
<http://bit.ly/subscribe>

Conference Attendance 201 - Learning While Confering, Continued - <http://bit.ly/vH7tgj>

Lessons learned in Android usability testing through programming

By Stefan Kläner

Motivation

Why would a tester start programming an Android application? Well, I got on a new project with a main focus on usability testing, so I started to research memory management, the Android API, UI design patterns and so on. As a Tester I like to be able to actively participate on design discussions. So I started programming simple applications, because in my opinion the best way to learn something is actually doing it. After an obligatory “Hello World”, which actually requires no programming at all, I needed something more sophisticated. While I was reading the Software Testing Club forum the idea was born, an application to read the latest blog posts of the Software Testing Club. But as I read What’s the best way to keep updated on the latest & on-going in Software Testing field? (<http://www.softwaretestingclub.com/forum/topics/what-s-the-best-way-to-keep-updated-on-the-latest-on-going-in>) I was reminded by what Robert C. Martin wrote in his recent book Clean Coder:

You should plan on working 60 hours per week. The first 40 are for your employer. The remaining 20 are for you. [...] I’m talking about 20 extra hours per week. That’s roughly three hours per day. If you use your lunch hour to read, listen to podcasts on your commute, and spend 90 minutes per day learning a new language, you’ll have it all covered. [1]

Reading the post changed my view about the application and I started to add more content to turn it into an application that provides Testers with current testing news with just a few clicks. So those people who are not willing to invest 20 hours per week, can at least have a lot of sources in one central place. Let’s assume the average home-to-office time is about 30min to 1 hour. So there are 7 to 14 hours a week that this application can make a little less boring. I have several ideas on how to improve the application, for example by adding some of the testing challenges Markus Gärtner created and/or collected, maybe some books that are available for free (The Little Black Book on Test Design by Rikard Edgren or Essential Software Test design by Torbjörn Ryber [as long as the authors are OK with it]), maybe adding the Carnival of Testers by Simon Morley or some testing cheat sheets. But I would love to hear what the testing community would add to the application and/or remove from it.

The source code is on GitHub (https://github.com/sklaener/android_stc), so everyone

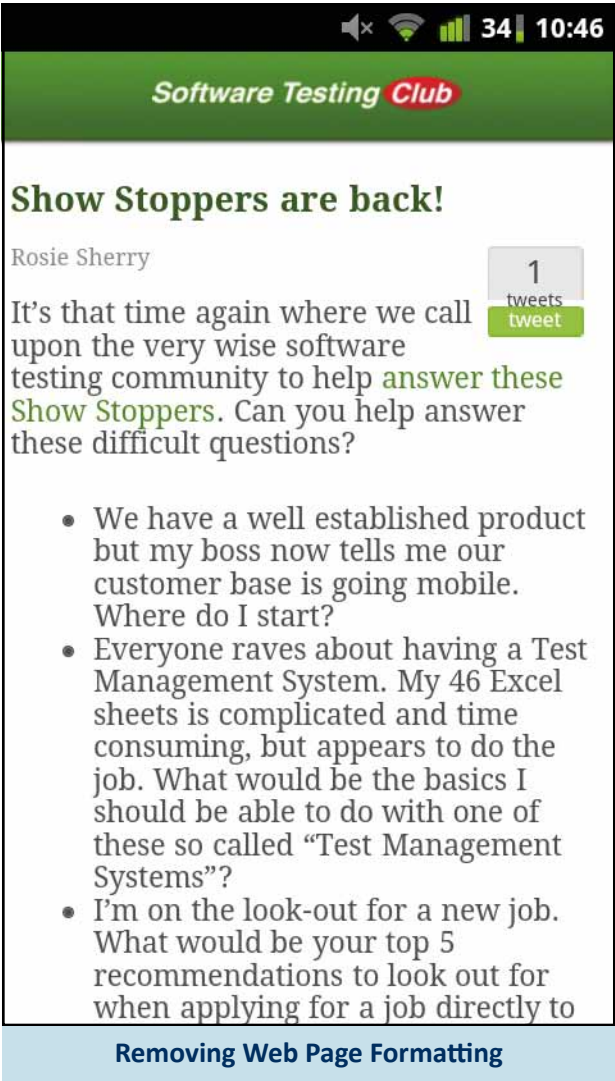


who wants to add something to the application is more than welcome to do so.

About the app

The first screen of the application is an implementation of the dashboard design pattern, and provides shortcuts to articles of the five different categories of the Software Testing Club blog (News, Community, Chronicles, Projects and Resources), to forum threads and to articles in The Testing Planet. In the top right corner of the dashboard are buttons to the STC Bloggers feed, podcasts feed and to the 20 newest videos. When pressing one of the buttons the content is displayed in the SlidingDrawer (which is somehow hidden on purpose) at the bottom of the screen. The menu provides all the additional RSS feeds provided by the STC, namely: Communities, Companies, Events, Jobs and also ten current or upcoming events. A long click on an item in the SlidingDrawer opens the URL in the default browser, a normal click opens the available content in a new view (Jobs, Podcasts and Videos are always opened in the default browser).

When a user clicks the comment button in



the menu (only available in article views) the article is opened in the default browser, as commenting through the application interface is so far not possible.

Lessons learned: Android usability

Remove web page formatting

The Internet provides a really great multimedia experience these days, but when reading an article the UI can destroy a lot of the experience. It should emphasize only the text on a page and not some fancy colored UI or flash animations. But even if flash is disabled or blocked, there will be spaces in the text where the flash content has been, which does not help to improve the readability either. So the only real option is to parse the HTML, remove the whole formatting and display only the parsed text.

Allow users to change font sizes

Android offers several ways to make text more readable, though they aren’t working really well. For example stock Android 2.0 (and higher) will re-flow

Continued on page 28





Continued from page 27

the text when you double tap, but not while pinch-zoom in and the re-flowed text is not necessarily more readable, especially when bullet points or something else are in the text. So pinch-zoom in is not really comfortable to use, but sometimes the only option to change the font size. But to zoom in on every article you want to read is quite annoying over time, so let the user choose a default font size, to make it a better experience. But don't offer choices like 'tiny', 'small', 'medium' or 'huge' as they only confuse the user, just use numbers.

Touch target sizes

A critical aspect in touch-based UI's is the size of controls. If they are too small, the user is frustrated while he tries to hit them and if they are too big the design wastes space for other features, so how big is big enough? The first thing we did on our project was to take a look at the user interface guidelines provided by Apple, Microsoft and Nokia (Google doesn't seem to have one). Apple recommends [2] a target area of "about 44 x 44 points" for tappable elements. Microsoft recommends [3] a target size of 9 mm (about 34 px) and a spacing of 2 mm (about 8 pixels). Nokia goes a step further and defines [4] a 7x7 mm target size with a 1 mm spacing for index finger usage and a 8x8 mm target size with a 2 mm spacing for thumb usage.

While it makes sense to have a bigger target size for thumb usage, the recommendations now vary between 7x7 to 11x11 mm, so the manufacturer recommendations aren't really helpful here. A study from the MIT [5] found out that the average size of a human finger tip is between 8 and 10 mm and the

average finger pad between 10 and 14 mm. So finally we had some information that was reliable and worth working with. In order to come up with the best possible target size for our users we created a simple application that had 8 buttons in total (4 placed in a line at the top of the screen and 4 ordered as a 2x2 square in the center of the screen). Every time a user clicked on a button he was directed to a page telling him which button he pressed and displaying two buttons (YES and NO) in order to verify that this was the button the user intended to click. We did this with a button size of 8x8, 9x9, 10x10 and 11x11mm and the results were actually quite surprising as the false hit rates were almost the same for 9x9 to 11x11 mm, but round about 2.5 times higher for a target size of 8x8 mm. So once again we learned to know what our customers want. Though in this case, asking a customer if he wants a 8x8mm or a 9x9mm target area is really not a good way, doing experiments in collaboration with some end-users is definitely the better way - for both sides.

Back button

Android devices provide a lot of buttons (either hardware or software), but no one on our team really uses them intensely, so we have not spent time thinking about integrating them properly and only checked if the default behaviour is acceptable. It turned out to be a huge mistake. Android uses a concept of activities and intent. Activities are basically what gets displayed on the screen, so e.g. the dashboard in the STC application is an activity (this is a bit over simplified, but it is not a programming article after all) and intents allow you to pass information between activities (e.g. passing parsed XML data to a WebView in order to display it). On the project we used a click on the logo - located in the menu bar on top of the screen - to jump back to the previous activity, but almost none of our users expected it to work that way, and we got a lot of complaints that the back button is not working properly. What they expected to happen was, that the back button not only shows them the previous activity - which is the default behaviour - but also to close menus, sliders and other on-screen overlays. So as simple as a back button sounds, it seems to be really important to most users.

Don't explain yourself (DEY)

More and more applications on the Android market contain a "how to" screen on the first start-up but let's face it, when users download an application they just want to try the app, reading instructions is the probably the last thing they want in that moment. So while I was programming the application, my goal was to achieve something that doesn't need to be explained. And every time I thought a "how to" screen might be helpful, I changed the UI to something simpler or more obvious. Google Goggles is probably a very good example, it is a great application but it introduces its features on multiple pages - quite annoying.

The problem of swiping

Swiping is probably the best way of navigating through a multi-page application, but as soon as it is combined

with other gestures it is very difficult to get it right. In the STC app swiping is combined with vertical scrolling and also with an OnItemClickListener. Combining horizontal swiping and vertical scrolling already introduces a lot of problems, because when a user scrolls down - and no user I know of scrolls down without swiping to the left/right a bit - the application should not interpret that action as a horizontal swipe, but the only options to make it a better experience are to change either the threshold swipe velocity or the minimum swipe distance. But to find good values, which allow the user to easily swipe and scroll down without swiping is not that easy, because you don't want to end up with a swiping mechanism that requires the user to swipe really fast, in order to have a better experience scrolling down. As the STC application also implements an item click, I really need your feedback if my current settings are good for the majority of the users.

When swiping is used, the application needs to visualize that there is more content available - so the user sees there is more information to reveal - and the easiest way to do this is using tabs. Tabs are working really great (especially on tablet's as they offer more screen space) but when more than 5 items are available it becomes messy. One possibility would be to show only three tabs at a time (previous, current, next) but in my opinion that destroys the whole meaning of tabs and being able to switch between them anytime - though the new Android Market app is doing it quite nicely. After trying different UI patterns I've decided to use dot indicators as they not only indicate that there is more information available, but they tell the user where he is in the content and they also tell the user how many pages exist. Another benefit of dot indicators is that they work on both tablet and smartphone, unlike tabs. □

AUTHOR PROFILE - STEFAN KLÄNER

Stefan Kläner works in the software testing field since 2008 with a main focus on usability and acceptance testing. Stefan is a firm believer in self education. When he is not testing he researches on systems theory, design thinking, sociology and psychology. He also occasionally tweets as @sklaener

REFERENCES

- 1 Martin, Robert C. (2011): The Clean Coder: A Code of Conduct for Professional Programmers. New Jersey, Prentice Hall
- 2 iOS Human Interface Guidelines http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/UEBestPractices/UEBestPractices.html#//apple_ref/doc/uid/TP40006556-CH20-SW1
- 3 User Experience Design Guidelines for Windows Phone [http://msdn.microsoft.com/en-us/library/hh202913\(v=VS.92\).aspx](http://msdn.microsoft.com/en-us/library/hh202913(v=VS.92).aspx)
- 4 Nokia User Experience Checklist for Touch http://www.developer.nokia.com/info/sw.nokia.com/id/cf2d9a8f-727b-437e-9378-c759ae335624/UX_Checklist_for_Touch.html
- 5 Dandekar, K., B.I. Raju and M.A. Srinivasan (2003): 3-D Finite-Element Models of Human and Monkey Fingertips to Investigate the Mechanics of Tactile Sense. Journal of Biomechanical Engineering, Vol. 125, ASME Press, pp. 682-691



Testers don't improve quality, those who listen to them do. @testertested



CHANGES SCRUM MADE TO THE TESTER'S ROLE

In the last decade, Agile methods went from quirky and novel to essentially standard practice. Scrum has been of the strongest proponents of the agile methodology throughout, but one question remains. How have methods like Scrum altered the tester's role in development organizations? Ulf Eriksson from ReQtest, the on the cloud bug tracking and requirements management solution, gives us some answers.

QR CODE LINK - WWW.REQTEST.COM



AUTHOR PROFILE - ULF ERIKSSON

Ulf Eriksson heads ReQtest, an online bug tracking software based in Sweden. ReQtest, which is the culmination of Ulf's decades of work in development and testing, is a very handy and simple tool to track bugs, list requirements and better manage all communication by anyone involved in any project. Ulf is also the author of many white papers and articles, mostly on the world of software testing. He is also slaving over a book, which will be compendium of his experiences. Ulf lives in Stockholm, Sweden.



TESTING IS CLOSER TO DEVELOPMENT

The Scrum philosophy dictates that a team should be as self-sufficient as possible, meaning that collectively, the team members should have every skill needed to complete a task. Since very few functions can be considered complete without testing, most teams need testers. In complete contrast to traditional development methodologies, Scrum pushed testing into development teams, both physically and procedurally. No other facet of Scrum methodology has had a bigger effect on the testing role than this one.

Traditionally, development and testing have been separated by an organizational chasm which persists even in some companies that have adopted Scrum. However, by planning and conducting testing in close conjunction with development work, these organizational boundaries are fading. Testers now enter the team earlier, very often at the same time as developers. Traditionally, testing tasks were an external activity consisting of planning tests, executing them on an on-going basis, and reporting results. Nowadays most testers sit scattered across development teams, and each team shares responsibility for software that is coded, tested, and ready for delivery.

Naturally, some testing activities should stay separate from development, partially to maintain objectivity but also because these activities come later in the sequence and hence can't be planned at the same time.



TESTING STARTS EARLIER

Scrum automatically includes testing in the development process as early as possible, which is undeniably positive and something that most organizations strive for. Only by testing can we review the business requirements early on, and then based on the testers' skills, influence the system design. This gets the testing discussion started early – even before the team starts writing code – while keeping testability from falling into the background where it frequently gets forgotten. Without this focus, teams often forget about testing during the requirements and design stages.

Defects, also referred to as bugs, design errors, or other terms, all signify something that's different from what is expected or required. The earlier you detect defects, the cheaper they'll be for you to fix. Correcting these deviations late in the cycle costs the team dearly, in the form of extra work such as additional documentation, regression testing, and potentially expensive re-delivery to the customer. Cost, stress, and embarrassment are all excellent reasons to introduce testing early in development, as well as to keep trying to discover errors as early and often as possible. There are countless cases where during an early requirements discussion, a tester's simple question made a client or coder rethink a critical point. Discovering defects at the design stage is considerably less costly than fixing failures found later.



ITERATIONS

When testing and development teams plan work together, the development team can deliver work more frequently for testing. If team members are allowed to hand over internal deliveries informally, they can spend less time on the detailed documentation normally required. Assuming that the team has flexible build and deployment tools, the development team can deliver work to testing frequently, with very small iterations inside the sprint. Of course, in a highly functioning scrum team, development and testing are not two separate entities, and are, for all intents and purposes, one, single development team, while in other team set ups, testers are not necessarily always involved.

Close collaboration between development and testing also creates opportunities for continuous learning within the team. Testers deepen their technical knowledge about the product being developed, making them more effective at planning the

testing strategy, and resolving the few bugs that will inevitably get through. Developers who are unschooled or inexperienced in testing can get a better grasp on how testing works, and can take better advantage of testing environments, testing data and other assets that the testing team possesses. Of course, in many cases, developers are highly attuned to the needs of testing, and in cases such as these, testing becomes an activity in which programmers and testers work together as a group and learn more about their respective areas, as well as others'.

Continuous delivery iterations are worth striving for, for several reasons. For starters, testers see software that's similar to previous iterations and isn't as "jumpy". Developers get feedback earlier, when they're still developing the software and can quickly correct bugs while they're actively working on the code. With repeated sprint iterations, the product owner also gets insight into how the work is progressing. By adding, removing, and prioritizing tasks from the product backlog, the product owner can continuously control how the product evolves.



THE TESTER'S JOB

The "purest" version of Scrum recommends that anyone on the team should be able to take on any role. In reality, this is very rare in medium to large-sized organizations because employees are usually either developers or testers, so teams typically have dedicated testers who are usually encouraged and even expected to ask tough questions. Doing so during design and development would have been overstepping boundaries even just a few years ago, before testing became well-rooted in the development process. Organizations traditionally expected testers to objectively examine the deliverables

Continued on page 30



Continued from page 29

they received, and report back on the quality (or lack thereof) they found.

In Scrum, testers are an integral part of the team, and provide feedback on pre and post delivered work in a close relationship with developers. Some teams even test on the developers' own machines so as to ensure basic functioning before even deploying to the test servers.

Scrum is occasionally said to introduce a new challenge to the testers' role: the risk that testers get too "chummy" with developers and don't fully expose the defects they uncover. While this may be true in a small number of cases, anyone will agree that it's certainly easier and faster to successfully explain defects or issues to someone face to face, rather than through an endless stream of emails. Personally, I find that the advantages gained this way far outweigh any risk of developers and testers covering up for each other.

5

THE TEST MANAGER'S ROLE

Among these broader changes, the test manager's job has shifted into more of a coordinating role, responsible for strategically planning testing across various teams. Testers take on tasks such as planning, reporting, and documentation in the team, and the test manager compiles and synthesizes their work. The test manager also has to coordinate different types of testing activities to keep major differences between teams from arising and collects and compiles test results on the team level. In large projects with parallel Scrum teams, it's easy for teams to focus solely on the one part of the system that they themselves are building, so the test manager has to ensure the integrity of the system, long before integration formally starts.

6

AUTOMATION

Agile methodology provides only the preconditions for shortening the time it takes to transform an idea into software. In Scrum, you expect that what the team produces in a sprint (1-4 weeks of development time) will naturally be high-quality enough to deliver to a customer, production, or end-users. This requires the testers to examine components newly developed in the sprint, as well as preexisting code that may be affected – which may be impossible without automated regression testing if the existing body of code is large. An additional advantage of automation is that it allows programmers to have the confidence to make changes as tests covering existing functionality are readily available. When combined with CI and overnight builds this approach can be crucial for really early feedback.

Testers should focus manual efforts on newly developed components, where it's most useful and is most likely to detect defects. You can save significant resources if the team of testers knows how to build automated test suites that can be continuously executed – even if it requires help from the developers. Of course this is not to say that the integration between components should not be seen as crucial, as often this could bring older and already tested areas to the spotlight when analysing how they now interact with other components within the system.

7

VALUE-ADDED ACTIVITIES

According to my perspective of Lean and Agile methodology, organizations should continuously seek to eliminate waste, so no one does work that doesn't add value. This is especially important from a tester's perspective, since testers, understandably, would prefer to have the project deliver smaller chunks of functionality with few defects rather than huge, bug laden releases. Project planners usually prioritize bug fixes over other tasks that the team needs to perform, but testers may be forced to tolerate defects going uncorrected, since the trouble to repair them simply does not create enough net value for the business. On the other hand, it can be less costly to the organization to make even small bug fixes with very little beneficial impact, if the developers simply perform them on their own initiative. If the team has to document, discuss, and then prioritize defects, they can waste hours just in administration tasks before they can even attempt a fix. This approach obviously requires that the project has come to a stage that allows "open season" on bugs.

What's next? - Scrum borrowed its basic principles from Lean. Read more in the article "Lean and Mean - Requirements management and testing that means business" in the ReQtest monthly newsletter - see link below. □

TESTERS' ACTIVITY FROM THE BLOGOSPHERE

Carnival of Testers

Oct '11

The word "continuous" pops up continually – all of the time(!) - whether it's integration, deployment, delivery or moaning(?) But one of the biggest tools that any tester has (or smallest if you're this guy [<http://www.simpsonstrivia.com.ar/wallpapers/homer-simpson-wallpaper-brain.htm>]) sits in our head and it aids continuous learning. Let's look at the learning reports, experiences and reflections from some 34(!) testers...

GENERAL

Trish Khoo has some thoughts on visual learning and mind maps. [<http://trishkhoo.com/2011/08/thoughts-on-using-mind-maps/>]

Zeger Van Hese on his illuminating learning journey with mushrooms and reflections to testing. [<http://testsidestory.wordpress.com/2011/08/17/finding-porcini/>]

For potential conference presenters here are some tips that Lanette Creamer has learnt. [<http://blog.testyredhead.com/2011/08/15/so-you-want-to-be-a-conference-presenter.aspx>]

James Bach on learning in testing and something called a Paired Exploratory Survey. [<http://www.satisfice.com/blog/archives/598>]

John Stevenson on learning aspects connected to product knowledge and exploratory testing. [<http://steveo1967.blogspot.com/2011/08/is-product-knowledge-essential-for.html>]

Learnings from the back of the room from Markus Gärtner. [<http://www.shino.de/2011/07/31/testing-dojos-from-the-back-of-the-room/>]

Rob Lambert on mind maps and

metrics. [<http://thesocialtester.posterous.com/mind-the-map>] & [<http://thesocialtester.posterous.com/are-we-halfway-there-yet>]

Lisa Crispin on learnings from conferences. [<http://www.eurostarconferences.com/blog-posts/2011/7/5/learning-from-conferences.aspx>]

Pete Walen on learning from engaging with other testers. [<http://rhythmoftesting.blogspot.com/2011/07/cross-pollination-or-how-talking-with.html>]

In the midst of all this learning - beware of fake learning! [<http://testers-headache.blogspot.com/2011/07/fake-learning-shortcuts-dont-always.html>]

Learnings from Ola Hytén from his ISTQB experience. [<http://testandtech.blogspot.com/2011/06/certification-experience.html>]

PEER CONFERENCES

THESE ARE WONDERFUL LEARNING OPPORTUNITIES

Brian Osman on KWST, here and here. [<http://bjosman.wordpress.com/2011/06/28/kwst-kiwi-workshop-on-software-testing/>] & [<http://bjosman.wordpress.com/2011/07/08/kwst-kiwi-workshop-of-software-testing-day-2/>]

Continued on page 31

READ THE REQTEST NEWSLETTER! - <http://eepurl.com/f-iZb>



"stories don't span iterations" Johanna Rothman. good read: ow.ly/7ce66 @karennjohnson

Continued from page 30

Lisa Crispin at WAT2, here. [<http://lisacrispin.com/wordpress/2011/07/04/writing-about-testing-2-style-and-grace/>]

Jeroen Rosink at DEWT, here. [<http://testconsultant.blogspot.com/2011/06/reflection-of-day-at-dewt.html>]

Markus Deibel at LEWT09, here. [<http://testingyet.markus-deibel.de/index.php?/archives/25-LEWT09-Takeaways-for-a-part-time-tester.html>]

SWET2 was covered by Torbjörn Ryber, Rikard Edgren, Sigurdur Birgisson, Ola Hyltén and Simon Morley – here [<http://thetesteye.com/blog/2011/04/thoughts-from-swet2/>], here [<http://thetesteye.com/blog/2011/04/highlights-from-swet2/>], here [<http://happytesting.wordpress.com/2011/04/13/swet2-and-triggering-words/>], here [<http://testandtech.blogspot.com/2011/04/swet2-great-way-to-spend-weekend.html>] and here [<http://testers-headache.blogspot.com/2011/04/swet2-serious-testing-talk-by-serious.html>].

Nancy Kelln and Lynn McKee

at POST, here [<http://www.unimaginedtesting.ca/blog/602>] and here [<http://www.qualityperspectives.ca/blog/2674>].

Markus Gärtner at GATE, here [<http://www.shino.de/2011/10/04/first-german-agile-testing-and-exploratory-workshop/>].

LEARNING ABOUT FRAMING AND TEST FRAMING

Michael Bolton on many key aspects. [<http://www.developsense.com/blog/2011/05/ive-been-framed/>]

Some insights from Michael Larsen, here. [<http://mkl-testhead.blogspot.com/2011/08/test-framing-or-helping-someone-make.html>]

Some learning from Jeroen Rosink, here. [<http://testconsultant.blogspot.com/2011/05/testframing-bit-of-my-perspective.html>]

Ben Kelly on aspects of Framing, here. [<http://www.testjutsu.com/framing-your-tests-framing-your-audience>]

NEW PENS

AS ALWAYS, A SHOUT-OUT FOR

NEW BLOGGERS - JUST STARTING TO DISPLAY THEIR LEARNING AND EXPERIENCES

Henrik Andersson makes some good parallels with conformity, here [<http://thetesteye.com/blog/2011/04/do-we-all-want-black-coffee/>]. It's reminds of the observation about certain teenagers, "the need for independence, especially if they can do it in a group..."

David Greenlees, here. on his takeaways and real learnings from conferences. [<http://davidstest.posterous.com/2011/04/true-value-of-conferences.html>]

Claire Moss, here, with an approach to attending conference tracks. [<http://blog.aclaification.com/2011/05/exploratory-learning/>]

A game analogy to learning about a product, here, from Daniel Berggren. [<http://outsidetheblackbox.wordpress.com/2011/02/14/exploring-the-world-of-software-testing-and-super-mario/>]

Daniel Snell, here, on his reasoning for starting a blog. [<http://dansnellsblog.wordpress.com/2011/06/03/why/>]

Jean-Paul Varwijk, here, with reflections from DEWT triggering the start of his blogging. [<http://arborosa.org/2011/06/17/dewtconferencepost1/>]

Conor Moore started his blogging odyssey, here, with some thoughts about ISEB. [<http://www.softwaretestingclub.com/profiles/blogs/iseb-intermediate-exams-and>]

First steps, here, from Nolan MacAfee. [<http://testerstrek.blogspot.com/2011/05/humble-beginnings.html>]

Huib Schoots took his first blogging steps, here. [<http://www.huibschoots.nl/wordpress/?p=1>]

More first steps, here, from Simon Schrijver. [<http://simonsaysnomore.wordpress.com/2011/08/28/good-start-for-a-junior-tester/>]

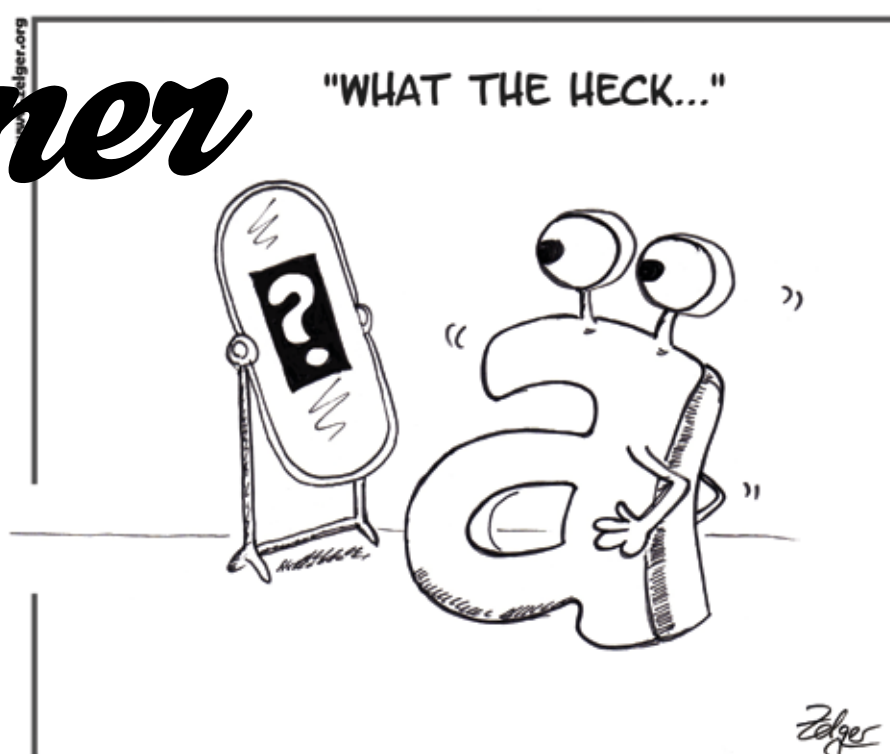
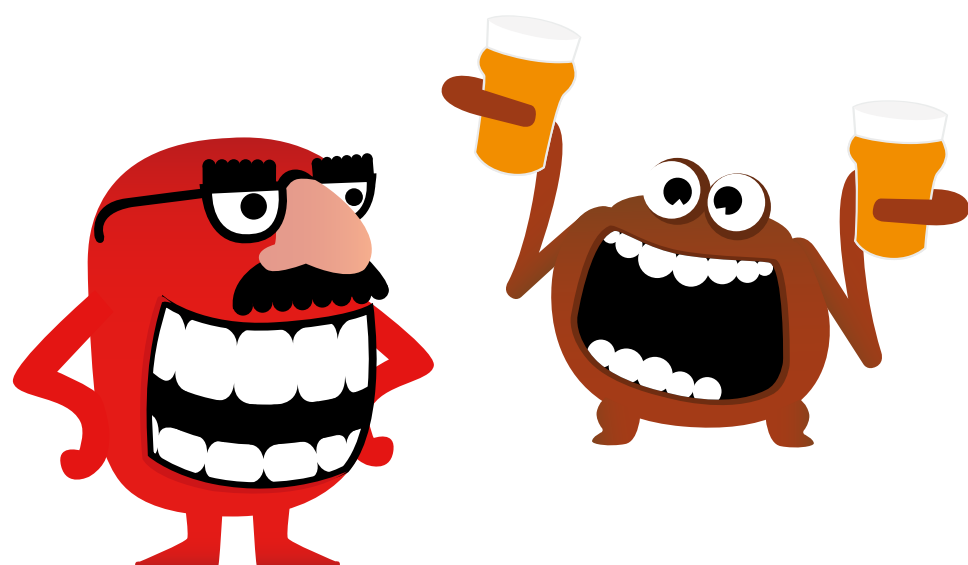
A meaty first post from Joris Meerts, here. [<http://blog.testingreferences.com/2011/09/26/my-four-favourite-articles/>]

THIS IS JUST A SAMPLE OF LOTS OF LEARNING GOING ON! HAVE YOU LEARNT ANYTHING? IF SO, HAVE YOU TOLD ANYONE ELSE ABOUT IT?

UNTIL THE NEXT TIME...

Simon Morley

the cartoon corner



All at once, the mirror did not recognize the German Umlaut anymore



It's all just testing - <http://angryweasel.com/blog/?p=349>



The Evil Tester Question Time

Provocative advice for testers who don't know what to do!

Q1. DEAR EVIL TESTER...

I want my developers to respect me and the work I do. I've tried learning the programming language they use but this takes time and effort. What's the easiest and/or quickest way to gain respect as a tester? - Andy

Dear Andy...

Thanks for asking this really important question. In fact it's so important I'm honestly going to give you real tools to deal with this situation. No joking around. No messing about. This answer provides "therapy on a stick" for the common tester.

I've heard the 'respect' cry since I started testing. I still hear this type of question asked at conferences and testing events so I'm sure plenty of testers are reading your question and nodding along. "Yup, I don't get none of that necessary respectful neither".

I too found myself caught up in that tester hysteria. I felt the way I imagine you might be feeling now. Poor poor pitiful me. But I got over it. I gave myself a healthy dose of sane juice to stop that delusional thinking. You can too.

But before I explain how, I need to mock you a little first. I don't want to mock you, but it's mandatory. I don't enjoy mocking you, but I took an

oath. Claiming 'respect' victim status means that I have to trigger your self-respect defence mechanisms. Think. Now. Have you got a skill? Any skill?

When someone comes up to you and says "Hey, I tried to learn your skill but I found it quite hard, it took too much time and effort so I stopped - do you respect me more now?" What's your first instinct? And mind your language.

Stop using that word. Respect is a woolly, ambiguous word. It spans the spectrum from "being nice" to "I'm almost in awe". You don't mean it. You don't want it.

Are you ready to defend yourself and tell me how my behaviour showed evidence of a lack of respect? Ready to tell me what you want me to do differently next time? [Wannabe life coaches note the use of provocation as a therapeutic tool]. And now. How.

How do you know they don't respect you? Did you ask them? "Do you respect me?" Did they say "No"? Ignore their answer; you'd still be no closer to your 'quick fix'.

The person you really need to ask is... you. How do you know they don't respect you? What, in their behaviour, do you use as the evidence that allows you to maintain the belief that they don't respect you?

That behaviour, or absence of behaviour, gives you an actionable request you can make of them or work with them to prevent happening.

If you know they don't respect you because... say... they don't invite you to the planning meetings Then you can say "I'd like to be invited to the planning meeting because then I'll know what's coming and can identify spiffingly splendid risks in advance".

[Note: It doesn't really matter what you say as a 'because'. You can get all tautological on them "... because then I can attend the planning meeting". Cialdini provides examples in "Influence - science and practice" [1] that Influence doesn't require a good reason, it just requires a reason.]

They might say "Oh, we didn't invite you to those because we thought the work you're doing was so important that we didn't want to waste your time." And then you know that you were delusional all along, and I saved you a small fortune on therapy.

Of course they might not. In which case, as well as asking myself "How?" I'll point out that I also learned how to juggle. Nothing pulls in the respect dollar like a good three ball clawed hand cascade.

Don't worry about payment; your tears of gratitude are payment enough.

All @ EvilTester.com

PS. always assume tears stem from gratitude.
PPS. delusions should help us not hinder us.

Q2. DEAR EVIL TESTER...

Why is there so much argument about the definition of "concurrent users" in performance testing? - Mark

Dear Mark,

The argument stems from the belief that people can't multi-task, so the notion of a concurrent user makes no sense to some people.

It remains my fervent belief that this argument would not take place if everyone learned how to juggle.

Hope that helps you in your next non-functional requirement meeting Mark,

All @ EvilTester.com

Q3. DEAR EVIL TESTER...

How do you prevent boredom while re-testing and ensure that you are always eager to break the tested product as you were the first day you got the product? - Mostofa

Dear Mostofa,

It might sound glib, but my immediate response is to say, "Stop doing the same things you did on the first day you got the product."

I must have a different attitude though, as I don't really understand your situation. If someone gives you something and asks you to test it again and again and again. How do your eyes not fire up?

Continued on page 33



Where are the technically inclined testers? <http://bit.ly/rUiizl>

Continued from page 32

How does your trident not tingle with delight? How could you not be eager to break the product? How is that possible?

The only time I don't feel that way is when I'm using software that I need. For example, I'm real careful when I use bank automated teller machines. A life lesson that cost me £50 when I couldn't afford it. True story, don't ask, the memories still pain me to this day. When I'm using, I try very hard not to break what I assume is a 'tested' product. When I'm testing. I'm not using.

But as to how? I'll tell thou, now. And this time I'll do it in rhyme. Because to not do so... would be a crime. Ahem...

AS IF IT WERE THE FIRST DAY

Testing's not besting the code they produce,
We work hard and challenge ourselves to deduce.
Ever more interesting ways that we might,
have deluded ourselves into thinking "everything's right".

If we ever glaze over and stop observing a new,
we'll miss bugs by the dozen not by the one or the two.

So partly its fear that keeps me in focus,
Fear that somebody else will notice, that tiny small thing of no real consequence
"How did I miss it, it makes no sense!"

There's no time like the past. To experience time is illusion.
There's no time like the present, so I have a solution.

I stay in the now, but not in the zone, I stay alert,
with intent, and I'm not alone
All of my skills, and my models and all of my tools,
the software can't win. I have no other rules.

I never re-test. I test and I test and I test and I test.
And now, again, as if for the first time, I will test it the best.

May all your good testing dreams come true,

All @ EvilTester.com

1 - http://en.wikipedia.org/wiki/Influence:_Science_and_Practice

Q4. DEAR EVIL TESTER...

How would you incorporate UAT into Scrum? - Zoe

Dear Zoe,

Scruatm? Are you just trying to trick me into writing a rude word? How dare you,

All @ EvilTester.com □

Top tips for your curriculum vitae

By Rob Lambert

What follows is my personal view of hiring Testers with some hints and tips on how to put forward an excellent CV to help get the job you want. The views and opinions will be different to other Hiring Managers, but I hope that some of the advice may help you understand a little more about how to put together a strong CV.

I believe in the capacity of all people to achieve great things. Yet I also know that some of these great things may happen in an environment different to the one I work in. There is a candidate for each and every organization. An ideal candidate maybe? Yet, if this ideal candidate can't articulate their skills and experience on their CV, or can't get their experience in front of a recruiter or Hiring Manager then they'll potentially remain elusive.

In today's social world it's becoming increasingly important to have a solid online presence too. I know of many recruiters who don't recruit outside of LinkedIn for example.

So you need to make your skills stand out. When I say stand out, I don't mean using pink paper for your CV. You need to make your CV the greatest advert of your skills. You need an online presence that helps to convey your message and add kudos to you as a candidate.

You need a Hiring Manager to be giddy with excitement at the prospect of the ideal candidate. Someone different to the masses; perfect for that Hiring Manager's environment.

What follows are some ideas from my perspective as a Hiring Manager. Obviously, what I look for in a candidate will be different to other hirers, but hopefully the following points may help to give you that edge, encourage to make some changes to your CV or to start thinking about how you convey your skills and experience in your application.

A disclaimer

This article will probably be completely irrelevant for those testers who are applying for a "resource" Testing role. What I mean by this is that the role you are applying for involves you ticking boxes and following scripts, and in essence, could be done by anyone.

The role is just another role. In these roles you will be treated as a resource (like a projector, or laptop) and standing out may actually go against you. Unfortunately, this type of role is very common, but we are seeing a tide of change.

The Basics

As with all communication, I would advise that you take the time to analyse the Audience and the Purpose. It should in theory be fairly straightforward to define the Audience and Purpose when thinking about your CV and online presence.

Audience

- Recruitment Consultant?
- Hiring Manager?
- Both?
- Who are they?
- What sort of business do they work in (straight laced, formal, chilled, startup, scripted, fun, modern)?
- How is their advert written (formal, brief, in-depth, informal, fun, old fashioned, boring)?

Purpose

- To intrigue?
- To persuade?
- To inform?
- To convince?
- To impress?

Once you have defined your Audience and Purpose you can then start to tailor your application, CV and online presence to fit these two elements. For example, if it's a formal job then a casual and fun CV won't cut it.

Look at what is expected of you in the role and tailor your application to suit.

A quick overview of what I think makes a great CV

- A document of no more than 2 pages. Website CVs are ok, but they are often not printable and many hiring processes require a submitted document
- An advert of your skills, experience and mind-set on Testing which will hopefully convince me that I really want to talk with you in person
- A short introduction to your previous career history
- A short summary of your main skills
- A short summary of what really makes you tick
- A low touch document to let me find out more about you (i.e., I don't want to struggle with formats, printing or language)

Continued on page 34



Continued from page 33

Most Hiring Managers want to read a simple, clean and superbly articulated CV that makes them keen to talk with you further.

I don't believe it should be a full-blown document that details everything you've ever done. If I want that level of detail, I will invite you to an interview or check out your LinkedIn profile (or other social presence).

It should provide a taste of what you are capable of, with just enough information to give me a good picture of you, your skills and your outlook towards Testing.

If I'm bored when I'm reading the CV, then you've conveyed a message to me about you; rightly or wrongly. Your CV should match your personality and the audience's expectations.

So how can you make your CV stand out?

Here are some top hints and tips on how to make your CV "stand out".

Make it match the job description

Everyone should have a stock CV that covers the basics.

This "Stock CV" should never be sent for any role you apply for.

A "Stock CV" sent as an application is easily detectable and suggests the person is just after a job. Any job.

The "Stock CV" should be the starting template for each new job application; the building blocks that enable you to keep your core message consistent.

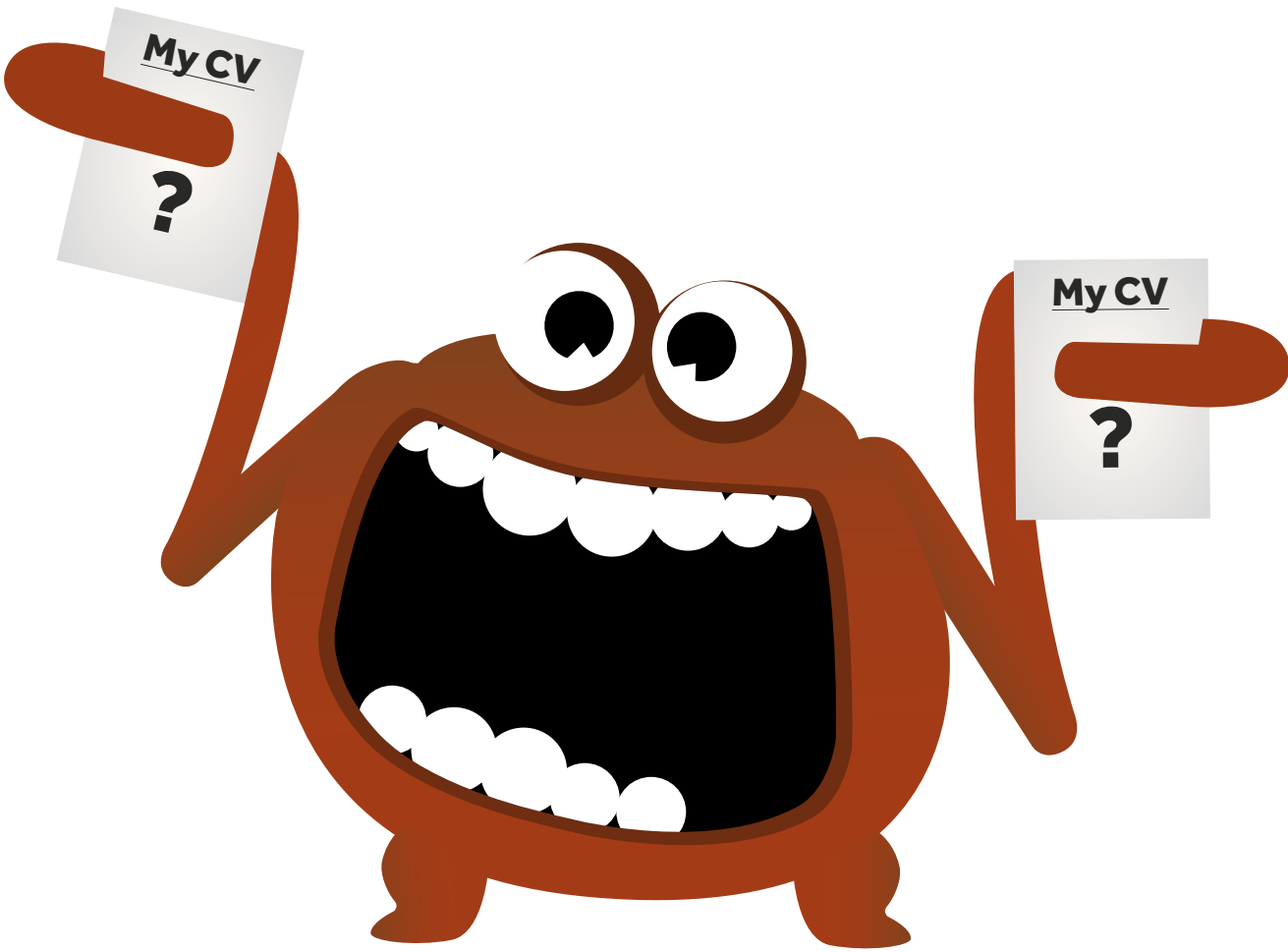
When recruiting, I look for people who want to work with me and crucially, work for the company I work for. This is very important to many companies who look for someone who will fit in, not just fit the skills set.

There is a reasonably healthy market for people who just want a job.

When I say make the CV match the job, I don't mean lie.

You need to think about the skills you have that meet the requirements described in the job role. It's a simple case of focusing on the skills that the Hiring Manager is looking for. It's not about lying or exaggerating, although an amount of "gloss" is to be expected. If you don't have the skills and really want the job then don't lie; instead spend your spare time developing the skills you need.

**ALWAYS
CHECK
SPELLING**



Writing a winning CV can be a daunting and difficult challenge

For example, if the role is asking for good SQL scripting skills, then focus on your experience with writing SQL scripts.

For example, don't describe in great depth how you configured a Linux Server if the role is asking for Microsoft Windows based skills only.

Don't just list Buzzwords

A giant list of Buzzwords is pointless for most Hiring Managers without some context. I could list a giant number of technologies and platforms and environments I've worked on, but the truth is I'm more comfortable in some than others. I have better skills in some than others. And I prefer working with some than others.

When you do list buzzwords, make sure they are relevant to the job role and give a little extra context about when you used these skills, how you learned these skills and any challenges you overcame.

Check your spelling

A spell check takes a few seconds to run but is well worth the effort as spelling mistakes on a Tester's CV gives a really bad impression.

Sure, there may be typos that are spelt correctly but are the wrong word. "There" and "Their" come to mind, but try to weed these out too, although I suspect these would be more tolerable than spelling mistakes.

If you are applying in a language that is not your native tongue then reach out to the community and see if anyone is available to help you fine-tune

the language or help you to review the application. There are always people willing to help you out.

Don't include a certification logo

I can understand why some people may want to include their favourite Certifier's logo on their CV when applying for a "Resource Role", but this is a real turn off for me. I have no issue with people taking the certification courses but it is just a short course with a certificate, not a branding for life.

Certificates will come and go. They are hotly debated and passionately discussed. They are not a sign of excellence. So I would advise not including the logo on your CV.

The list of certifications belongs in the career development / training / education section.

Why include a logo of your certification board and not one of your University or School or Gym Club or favourite Operating System?

Create and nurture an online presence

I would expect every CV in today's market to include a section listing your online presence.

At a very basic level, this should be a LinkedIn account. At a more advanced level, it may include Twitter links, a blog or a website.

I'm not suggesting you MUST have an online presence, but I would expect any practicing

Continued on page 35



Continued from page 34

Tester to have some form of online community involvement. Maybe you belong to the Software Testing Club or another online community, or you contribute to an Open Source project or do some voluntary testing for charitable/non-profit organisations? These all add a level of commitment, experience and enthusiasm that's tough to get across in a CV.

It's a definite plus to include these channels on your CV, and here are two good reasons why:

- Recruiters are increasingly looking at LinkedIn profiles of candidates to garner more information before inviting for an interview or sending a CV to a client. (If you're not LinkedIn, you're increasingly Left Out ...or something like that)
- Interviews (even phone screens) are expensive and time consuming, which is why many people search for as much information beforehand as possible to aid in the decision making.

I like to think of your online presence as being an extension of yourself. It can provide Hirers with a great deal of information about what you think, what ideas interest you and what you do to keep your skills sharp.

I've always worked on the assumption that I would rather be open and honest about my online presence than let the Hiring Manager search for me and potentially find someone else; someone who may not be the greatest advert for me.

Saying that, it is also important to point out that a Hiring Manager would be negligent to hold the past against you for a current role. However, there are countless tales of people being fired because their management team found something offensive online that they wrote 10 years ago. It happens and it's becoming increasingly problematic for generations growing up with the social web at an early age.

So be careful, but don't be put off. An online presence really can make the difference to your application. A positive online presence will communicate much more than a CV alone could ever hope to.

Have a section outlining your past work and projects

A section containing all of your last work and projects is a good idea, but only if it's summarised to highlight the key points. Many people will have

AUTHOR PROFILE - ROB LAMBERT

Rob Lambert is The Creative Director of The Software Testing Club as well as the Test Manager at NewVoiceMedia where he's helping to build an awesome team of Testing talent. He normally hangs around Twitter here @rob_lambert and can be found blogging at The Social Tester. [1]

too many roles to list on a CV so I'd suggest you summarise to make it relevant for the role.

You may wish to supplement it with a website that provides a full listing. I have no qualms about reading more on a web site. In fact, I'd be positively impressed.

The problem with including everything in great detail in your CV is that it could start to consume the whole CV. A large amount of this career history listing could be repetitive and may add little value.

I'm personally interested in what you are like now and how you have developed over the last few years. What you did 10 years ago has surely influenced who you are now, but do you currently hold the magic I'm looking for right now?

Don't get too hung up with "years of experience". It is better to have three amazingly diverse and creative years than ten years of the same year. Experience is hugely important, but so too is relevance to the role, to the culture and to the fast-moving industry.

Make your CV look good and print it out

A massive list of bullet points is a big "no no". It's always good to add easy to read and descriptive titles to sections and bullet points are to be encouraged, but think about how easy the text is to read. Make it as simple and easy to read the highlights as possible.

In my experience, many people read documents by skimming the headings and then skimming the first few sentences within each heading before deciding to read the sections in full. Play to this technique by adding clear sections, clear titles and well written, but simple sentences.

I saw a Hiring Manager once do this skim review and create three piles of CVs. One pile was for YES to an interview. One was for NO to an interview. And the other was a MAYBE about whether or not to read it further. He would only return to this pile of MAYBEs if no one in the YES pile were good enough. A harsh approach, but when faced with 200+ applicants it's a logical system to avoid wasting time.

So even if you have the skills, you still need to articulate them clearly and get yourself in the YES pile.

The clearest way to do that is by using a neat layout, short and simple sentences that are easy to read and as little jargon/technical language as possible. Or at least use this approach for the first few sentences, which will lead people in to the finer details if they are interested, which of course you want them to be.

Appreciate that your CV will be printed out, so check it looks good and is still readable when printed. Check your page margins too. It would create an awfully bad first impression if the Hiring Manager had to edit your CV just to get it to print properly.

Don't use colour to signify anything

Not only is about one in twelve people colour blind, but also more often than not CVs are printed out in black and white. Any meanings you intended using

colour could be lost, which could make your CV appear jumbled or confusing.

Watch your language

Most Hiring Managers (if they are looking to build a great team) will be looking at how well you will fit in with the team. A person fit is often more important than a skills fit. You may be awesome at Testing, but if no one likes you and you cause grief in the team then that's a problem.

Watch your language in the CV. Study it. Re-read it. Put yourselves in the shoes of the Hiring Manager. Read about good language use on the web. Look for elements of language where you may appear selfish and not a team player. Or you may describe your skills in a way that shows a lack of confidence. You may come across as bullish or aloof or arrogant.

There are plenty of good books on language, writing and communication. "Drop The Pink Elephant" by Bill McFarlan is a good start [2].

Leave the Jargon out of it

Unless you are applying for a role that asks for specific tools, try to keep your jargon use low.

For example, a well-known mainstream tool uses the term CR for defect (or issue, ticket, etc.), yet in some businesses a CR may refer to a live architecture change or requirements change or something altogether different.

I'm not advocating ISEB/ISTQB standardisation, as universal Testing terms are a myth. But if you must use tool talk, then it's always best to explain what that term means and lay down the context so it makes sense to the reader.

For example:

"When raising a CR I always include a good title, main steps to recreate and version numbers". This is fine, but it would be much better as "When raising a CR (defects/bugs) I always include a good title, main steps to recreate and version numbers"

Using Jargon and/or language very specific to your environment can be confusing and misleading for the reader. Always consider whether your language would make sense to someone who didn't work in the same environment.

However, I have made a massive assumption with this point.

I am assuming that every Tester knows there are different terms for the different elements of Testing. If you really don't know or can't think of a way to describe your element/idea, then a few minutes spent on a Testing forum should give you some idea about what terminology others use or how best to explain it.

A sample CV

Here's my opinion of what I feel would make a good CV in terms of sections and content, obviously the following would be useless if the content inside is not worthy.

Continued on page 36



Continued from page 35

Current Employment
Explain your role, responsibilities, challenges, learning, day-to-day activities, tools and techniques. Think about how you work as a team, how you push boundaries, how you help deliver great software.
About Me
Your address, telephone, contact details, blog/web/twitter information
Testing Information (or similar title)
Describe your outlook on testing, automation experience, testing experience, skills, etc. that are relevant to the role. How do you up-skill yourself? What communities are you involved in? Do you volunteer your spare time to test Open Source products? How do you address current Test problems like cross browser testing?
Education
Your education background. I find this section interesting because my job descriptions never

include formal education backgrounds (school, college, university etc). I'm more interested in someone with passion and enthusiasm and self-learning. Saying that, it's a de-facto standard to include this section and some employers still cling to formal education as a filter technique.
Training
In this section, I would include all training completed including certifications. No logos though. Date of Training Training Skills obtained
Tools and Applications
An opportunity to describe your skills and experience with tools and applications. For example, how competent you are with Selenium or Firebug.
Previous Employment
I would list each company with relevant dates, complete with a "one-liner" on what you did there, what skills you used and what you achieved.
References
Include suitable references here or state they will be available if successful for the role.

Conclusion
Your CV is often the first impression your possible future employer has of you and you have to make it count. However, no amount of prettifying a CV or using simple language will work if you have nothing of value to say. If you do feel you have nothing to shout about and want to experience more of the software testing world, then hop on board to The Software Testing Club [3], join some Testers at the Weekend [4] or on a Weeknight [5], volunteer your time to Open Source or Non-Profit products, or simply start taking part in the big wide world of the Software Testing Community. Oh yeah, and if you are looking for your next Testing role, then why not check out The Software Testing Club job board? [6] □

REFERENCES
1 http://www.thesocialtester.co.uk 2 http://www.amazon.co.uk/Drop-Pink-Elephant-Ways-Mean/dp/1841126373/ref=sr_1_1?s=books&ie=UTF8&qid=1318935946&sr=1-1 3 http://www.softwaretestingclub.com/ 4 http://weekendtesting.com/ 5 http://weekendtesting.com/archives/tag/weeknight-testing 6 http://jobs.softwaretestingclub.com/

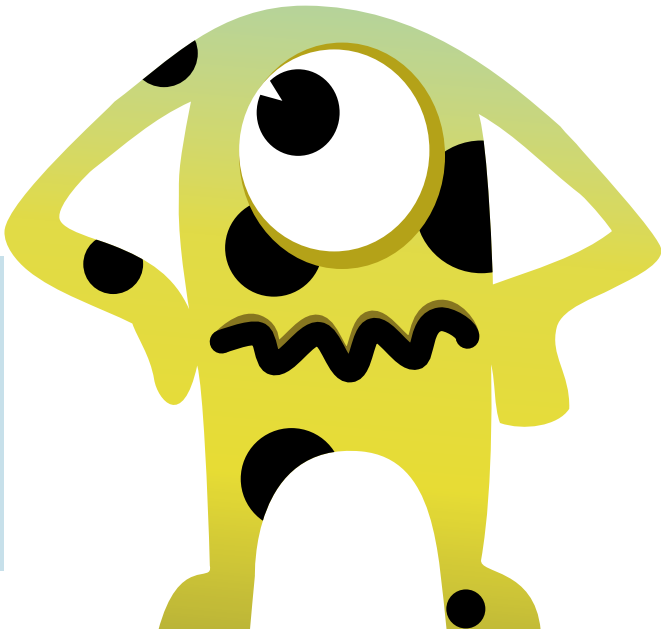
STC JOB BOARD | STC TESTING EVENTS | EVENTS @ SKILLS MATTER

LATEST JOBS
Editor of The Testing Planet - STC - Remote
Test Analyst - Hill Newton - Remote
System Test Analyst - Extol IT - Reigate and Banstead
Agile Tester - Brightpearl - Bristol
Test Analyst - Hill Newton - Remote
Senior QA Engineer - Linguamatics - Cambridge
Test Automation Consultant - michaelnomad - Brighton
Test Engineer - Veritech Infosystems Pvt Ltd - Kuala Lumpur
Test Lead - Veritech Infosystems Pvt Ltd - Remote
Agile Test Analyst - Star Q - London
Technical QA Manager - Webcrowd UG - Berlin
For a full list of jobs, visit... http://jobs.softwaretestingclub.com

SOFTWARE TESTING EVENTS
Chicago Software Testing Club Meetup - Nov 10th 2011 http://www.meetup.com/SoftwareTestingClub/events/36080482/
Birmingham Software Testing Club Meetup - Nov 15th 2011 http://www.meetup.com/SoftwareTestingClub/events/37167892/
TestBash - March 2012 www.ministryoftesting.com
Rapid Software Testing with James Bach - London March 7-9th 2012 www.ministryoftesting.com
Exploratory Testing with James Lyndsay - Cambridge May 2012 www.ministryoftesting.com
Join the Software Testing Club Meetup group today! http://www.meetup.com/SoftwareTestingClub/
For a full List of Events, visit... http://softwaretestingclub.com/events

EVENTS @ SKILLS MATTER
Where is the Quality in Agility? - Nov 18th 2011 - FREE http://skillsmatter.com/podcast/agile-testing/quality-agility/wd-2795
ThoughtWorks Continuous Delivery for Dev/Ops - Nov 16th 2011 http://skillsmatter.com/course/agile-testing/thoughtworks-continuous-delivery-for-devops/wd-2795
Uncle Bob's Advanced Test Driven Development - Nov 28th 2011 http://skillsmatter.com/course/agile-testing/uncle-bobs-advanced-test-driven-development/wd-2795
Robert Schneider's Mission Critical Service Testing Using soapUI Pro - Dec 1st 2011 http://skillsmatter.com/course/agile-testing/soapui-pro-training/wd-2795
Gojko Adzic's Test Driven Development Workshop - Jan 16th 2012 http://skillsmatter.com/course/agile-testing/gojko-adzics-test-driven-development-workshop/wd-2795
For a full List of Events, visit... http://skillsmatter.com/course

Test case management software comparison chart



Every software testing project needs to be organised. And if you really want to be efficient some kind of system is needed to manage all the requirements and data. Once an excel spreadsheet becomes too much for your test team to manage, you may want to start looking into a Test Management Solution.

Here we have selected 4 leading and forward thinking vendors to help you make a potentially very important decision!

	TESTUFF	PRACTITEST	TESTRAIL	REQUEST
Price	\$20pm / \$220 yr (per user)	From \$35 per user, per month	From \$239/user or less, \$25/user/month or less	€5.00-30.00 per user, per month
SaaS or Installed	SaaS	SaaS	Both	SaaS
Configurable	✓	✓	✓	✓
BUG TRACKING				
Integrated Bug Tracking	✓	✓	-	✓
Integration with other bug trackers	✓	✓	✓	-
Screenshot capturing	✓	-	✓	✓
Video Recorder	✓	-	-	-
Email Integration	✓	✓	✓	-
Email Alerts	✓	✓	✓	✓
Test Timer	✓	-	✓	-
TEST EXECUTION				
Automation Integration (API)	✓	✓	✓	-
Task/Test Management	✓	✓	✓	✓
Export Test Cases	✓	✓	✓	✓
Reporting	✓	✓	✓	✓
Statistics & Graphs	✓	✓	✓	✓
Test Plans & Configurations (e.g. browsers, OS, etc)	✓	✓	✓	✓
Archive & Audit Past Test Results	✓	✓	✓	✓

TestRail

bit.ly/stctr

PractiTest

bit.ly/stcpractitest

ReQtest

bit.ly/stcrequest

teststuff

teststuff.com

TESTUFF.COM - GET 10% OFF UNTIL 31 DEC 2011 - PROMOCODE: STC3112

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory

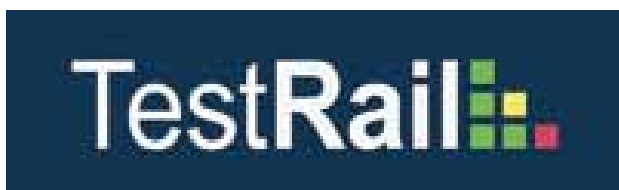
TEST TOOLS & SOFTWARE



GEMINI

GEMINI

Gemini brings versatile test management, bug and issue tracking to your team. Sign up to our cloud-based offering or install locally. Join the new generation in software project management with Gemini – no hidden extras or crazy pricing. 3 Users FREE – No Gimmicks – Full Edition. www.geminiplatform.com



TESTRAIL

TestRail – Test Case Management Software for QA and Development Teams. Comprehensive web-based test case management software to efficiently manage, track and organize your software testing efforts. www.gurock.com/testrail



PRACTITEST

Practitest is a SaaS-based Test Management Solution that supports the entire QA lifecycle, including requirement & issue tracking. www.practitest.com



REQTEST

ReQtest is an easy to use bug tracking software, available in the cloud 24/7. It empowers teams to work more efficiently and gives decision makers meaningful data on progress made. ReQtest includes a requirement management module which is tightly integrated with the bug tracking features. www.reqtest.com



KALISTICK

Kalistick gives testers a new solution to design efficient test strategies focusing on business risks. Our unique technology analyzes test cases footprints and functional changes to select the most relevant test cases. Discover how to move one step ahead in testing efficiency. www.kalistick.com



BUG DIGGER

BugDigger removes the hard work from web site bug reporting. With the help of a browser add-on, automatically captures and uploads: – web page screenshot optionally annotated using built-in editor, – environment details, and – web site usage history. Even busy or inexperienced testers can create useful bug reports instantly. BugDigger integrates with JIRA, Basecamp, Pivotal Tracker, FogBugz, Unfuddle, Redmine and others. www.bugdigger.com



PARASOFT SOATEST

Parasoft SOAtest automates web application testing, message/protocol testing, cloud testing and security testing. Parasoft SOAtest and Parasoft Load Test (packaged together) ensure secure, reliable, compliant business processes and seamlessly integrate with Parasoft language products (e.g., Parasoft Jtest) to help teams prevent and detect application-layer defects from the start of the SDLC. Moreover, Parasoft SOAtest integrates with Parasoft Virtualize to provide comprehensive access to traditionally difficult or expensive to access development and test environments. Parasoft SOAtest provides an integrated solution for: End-to-end testing, Environment management, Quality governance, Process visibility and control. www.parasoft.com

TESTPLANT

TestPlant develops eggPlant the leading user interface test tool that creates an abstraction of a GUI for any device type, enabling automation of screen-based testing through 'search and compare'. Download now. www.testplant.com

XSTUDIO

XStudio is a free ALM/test management solution allowing to manage requirements/specifications, scrum projects, Automated/manual tests, campaigns and defects. An LGPL SDK is also included to interface with proprietary tests. www.xqual.com

TESTLODGE

TestLodge is an online test case management tool that allows you to manage your test plans, requirements, test cases and test runs with ease along with issue tracker integration. www.testlodge.com

TESTOPTIMAL

TestOptimal – Model-based data-driven test design and test automation to improve test coverage, enable rapid response to changes and reduce test maintenance cost. www.testoptimal.com

LOADSTORM

LoadStorm – The lowest cost and easiest cloud load testing tool. Free account for 25 users. Test up to 100k vusers. Real-time graphs with key performance metrics. www.loadstorm.com

SOFTWARE TESTING TRAINING



SKILLS MATTER

Skills Matter supports a community of 35,000 Software Professionals with the learning and sharing of skills to write better software. Find hundreds of meetups, talks, conferences, skillscasts and workshops on our website: www.skillsmatter.com

SOFTWARE TESTING SOLUTIONS



ELECTROMIND

ElectroMind offers training, consulting, coaching and mentoring services to the software testing community. Through strong relationships with world-class testing experts, built up over several years, ElectroMind delivers niche training products, test process improvement consultancy and innovative people skills development programmes. Our

Continued on page 39

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory



Follow us for the latest and greatest news @testingclub

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory

Continued from page 38

consultants are comfortable using both traditional and Agile testing methodologies with experience in several industry sectors including financial services, telecommunications, online retail, travel, mobile and digital media. Through strategic partners, ElectroMind can offer performance engineering services including load and stress testing. Our overall philosophy is simple. We believe your software quality matters. www.electromind.com



TEST HATS

Test Hats are an independent software testing services provider, with offices in the UK and Spain. We provide a full range of testing services including System, Performance and Security testing along with specialised Consultancy and Training. For near-shore testing our Test Lab is fully equipped with a range of desktop and mobile platforms, testing software and tools, allowing us to provide a quality service at a competitive price. Visit our website to learn more about Test Hats and our services. Get in touch today to talk about how we can help test your projects. www.testhats.com



THE TEST PEOPLE

The Test People delivers the best, most innovative, highly technical and competitive performance engineering and test service available today. Based upon our extensive experience, TTP can deliver tailored services to address all aspects of the functional and non-functional test lifecycle, including highly specialised performance engineering and test automation services including automated build and continuous integration solutions. TTP are at the forefront of utilising the cloud for test and load environments, with significant experience in open source and the major commercial toolsets whilst also coming armed with our own performance and automation frameworks. www.thetestpeople.com

ORIGINAL SOFTWARE

Original Software - With a world class record of innovation, Original Software offers a solution focused completely on the goal of effective quality management. By embracing the full spectrum of Application Quality Management across a wide range of applications and environments, products include a quality management platform, dynamic manual testing, robust test automation and test data management. More than 400 organisations operating in over 30 countries use Original Software solutions. Amongst its customers are Coca-Cola, Unilever, Barclays Bank, HSBC, FedEx, Pfizer, DHL and many others. Visit www.origsoft.com/solutions for more information.



brainual software testing services

MOOLYA

Moolya is a new generation software testing services company headquartered in Bangalore, India founded in 2010. Our focus is to help business move forward. We believe in helping our clients to make great products that wow their customers. That's when we win. We are context driven testers highly skilled at exploratory testing, SBTM, small "a" agile testing and check automation. How can we help you win smiles on your customer's face? sales@moolya.com / www.moolya.com



REVOLUTION IT

Revolution IT is the leading Quality Assurance and Testing, management consulting firm in Asia Pacific. We help our clients deliver IT projects and have core offerings across Project Management, Requirements Management and Application Testing. We have over 250 staff and offices in Melbourne, Sydney, Brisbane, Canberra, Adelaide and Singapore. Our offering includes delivery consulting, methodologies, tool solutions and training. We have strategic partnerships with HP software, IBM Rational, Oracle, Agile Academy and

SAP. With HP we have been the leading HP Software Platinum Partner for 4 years running and the leading reseller, 1st line technical support, training and services partner. www.revolutionit.com.au

INDEPENDENT TESTERS, TRAINERS AND CONSULTANTS



ANNE-MARIE CHARRETT

Anne-Marie Charrett is a testing coach and trainer with a passion for helping testers discover their testing strengths and become the testers they aspire to be. She offers a blend of online coaching and training on Exploratory Testing, Career Management and motivating testers. Anne-Marie is currently working on a coaching testers book with James Bach, due out late next year. www.testingtimes.com.au

COMMUNITIES, CONFERENCES AND NEWS



SOFTWARE TESTING EUROPE

Software Testing Europe – Job Board, Training and News for Professional Software Testers in Europe. www.software-testing-europe.com



EUROSTAR

EuroSTAR is Europe's premier software testing event and will be taking place this year in Manchester, UK from November 21 – November 24. At EuroSTAR 2011, the leading names in testing will meet for an intensive 3-4 days of learning, networking, discussion...and a few extracurricular activities! Attendees can choose from numerous thought-provoking presentations, intensive tutorials, interactive sessions and inspirational keynotes. Plus, visit Europe's largest software testing exhibition which will be showcasing the leading companies in the industry. The EuroSTAR Team hopes to see you in Manchester later this year for what will be a fantastic, fun and innovative conference! www.eurostarconferences.com

THE TESTING PLANET DIRECTORY - GET LISTED WITH THESE AWESOME COMPANIES - thetestingplanet.com/directory



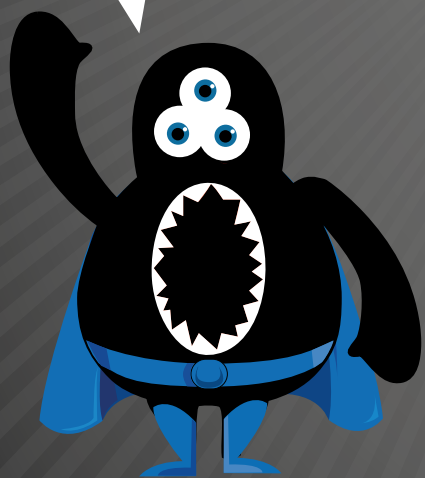
Thank you for reading. If you valued this, please pass it on. Till next time!



MINISTRY OF TESTING

CO-CREATING SMARTER TESTERS

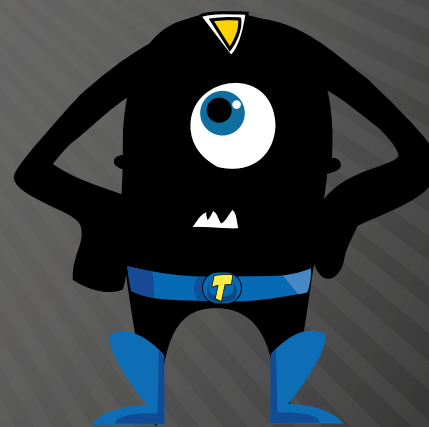
EDUCATION



COLLABORATION



EVENTS



WWW.MINISTRYOFTESTING.COM