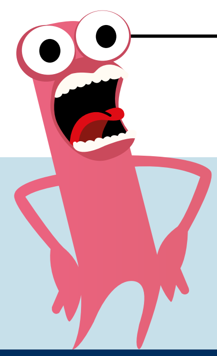
THE TESTING PLANET

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 effectiveness 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

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.kiwanja.net/blog/2009/01/a-glimpse-into-social-mobiles-long-tail/),and users need a healthy applications ecosystem to dip into, whoever and wherever 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 motivates scientists is a lack of info rather than the presence of info. Which is also true for testers. #thecanon #testing - @ruudcox