Just ban just!

30 January 2011 · exploratory testing, just, pre-scripted testing, regression testing

Office meetings are interesting events. Seriously, even the most boring ones. There's usually an important reason for the meeting, even if that meeting has been lost in the sands of Outlook repeat-booking, or the present attendees feel strongly otherwise. There's usually some kind of agenda, although as the meeting progresses, you may suspect the real agenda is hidden. But none of that is what captures my imagination. What sticks in my mind is the perspectives of the people in the room, especially when they are talking about a subject close to my heart like testing. We can't help but give away our positions and perceived hierarchies when discussing what work we need to do, and how.

For example have you heard someone utter, in a meeting, words to the effect of: "Just create a fully automated framework for our regression test pack, to test the future releases." Then continue on, as if they had asked to "borrow a pen", or if you could "just adjust the projector height". Before you have recovered from that salvo, you might be hit with the follow up: "can be 'shipped' next week? For the end of the sprint". If, like me, your comprehension of the practicality of software testing is different to the speaker's, you might have trouble answering.

As we're not in a meeting, and not surrounded by nodding heads looking sternly our way. Now, we have time - we can take a moment to reflect on that exemplary sentence. It's exemplary in that it gives us a great example of a mindset and viewpoint from which many people view software testing. A viewpoint which sees software testing as a simple process of building simplistic confirmatory checks into a suite of 'write once use forever' tools.

Let's break the above command down into parts and rewrite a little less concisely. I suspect a great deal of the hard work and failure is hidden behind those familiar words.

"Just", in this context probably means 'merely' or "nothing more than". They are suggesting the task is trivial. OK, so lets translate that to: Merely.

"Create", we know this system is a machine and probably software. As such we'll go through the usual steps for trying to create 'good software': Program, build, test & fix and release [a production quality system]. To avoid the risk of opening up a recursive program->automated-test->program->automated-test infinite-loop-shaped rip in the fabric of space-time: I'll assume this 'created' system is built without test automation, as it's described here.

"A fully automated framework", we might keep 'fully' as we know it means 'complete' or 'everything we are concerned with' in this sentence. But the word "Automated" in popular usage would suggest such technologies as autopilots for aircraft and yachts or the robotic production lines that were destined to replace car-factory workers. Essentially they want it to operate the software under test without human interference, indefinitely. As such it's going to

need to handle the usual problems that crop up in the use and abuse of software. Much as an autopilot system might correct for cross-winds, our system will need to handle changes in performance, wording, ordering etc. Notice, They haven't asked for autonomation or some other human/machine integration, An approach that might balance maintenance with automation. As for 'framework', we'll assume that's inline with typical development meanings. That is a toolset and/or API that the testers could use or override when creating their 'tests'.

"Regression test pack", I'll assume 'regression test[ing]' is intended to mean the same as the Wikipedia definition (http://en.wikipedia.org/wiki/Regression_testing), that reads: "software testing that seeks to uncover new errors, or regressions, in existing functionality after changes have been made to the software"

"Test", That can have a few different meanings, If we abide by Michael Bolton's differential description of testing versus checking (http://www.developsense.com/blog/2009/08/testing-vs-checking/) then test has a meaning similar to this definition in the Shorter Oxford English dictionary:
test [verb(2)] 2. verb trans. Subject (a person, thing, theory, etc.) to a test; try (out); evaluate (a hypothesis etc.) by experiment or critical examination; put to the test; try the patience or endurance of (a person).

"Future releases", as we are agile and iterative, this is part of the deal. The orator included this, I presume to ensure that the system was not so brittle as to become ineffectual or worthless as the system was added to. Caution here should ensure that the system is low or no maintenance. Note the maintenance cost requirements, or the Total Cost of Ownership, are not mentioned my the speaker, maybe they don't realise the costs involved.

Above we've worked through the key words in the sentence, providing a little more detail to the phrase. I have of course, made many assumptions. This is intentional, those stern faces sat around the table have probably made similar assumptions. They may have heard the same words you did, but didn't see the messy detail that we have roughly translated as:

"Merely, Program, build and test and fix and then release a production quality system capable of operating complex software - fully independently, That can find newly introduced failures in any part of the existing systems that, tests, uses and questions the new system and all its future implementations."

This is where you manage expectations. For example, Just or 'Merely'. We know this exercise is not trivial, and is likely to take many man-hours if it's possible at all. Lets start identifying the complexities that mean it won't be 'just' done immediately. Just is a tremendous understatement here. From now on, lets just ban 'just'.

Also the word 'Test' needs clarification. Let everyone know that test-automation isn't automated-testing. James Bach's Manual tests can not be automated (http://www.satisfice.com/blog/archives/58) post is a place to start. Come up with some examples of what people think is

being tested and maybe highlight what isn't actually being tested in those automated regression tests. E.g.: Question: We auto-test the arithmetic feature, right?

Answer: Actually we check two additions, one subtraction and a divide by zero… in one browser only. No big numbers, negatives, no division, no floating-point numbers and no Internet Explorer 6 & 7 users. We don't do a visual inspection - for example - to see if the text not being obscured by the advertisements. We don't actually use the calculator's arithmetic function ourselves…

Further to this are the problems with regression testing in general that I've discussed before (http://www.investigatingsoftware.co.uk/2010/12/arrogance-of-regression-testing.html). Spending too much time on existing regression 'tests' alone can distract from other existing issues and new emergent ones.

As a tester you can let the people in the meeting know the details, the messy truths. Inform them of the limited capabilities of the software. I don't only mean their new system, but the finite list of what can be automated reliably and effectively in software. Dispel their myths and undermine the test dogma: That is why you were hired.