Scatter guns and muskets.

30 January 2018

Many, Many years ago I worked at a startup called Lastminute.com (a European online travel company, back when a travel company didn't have to be online). For a while, I worked in what would now be described as a 'DevOps' team. A group of technical people with both programming and operational skills.

I was in a hybrid development/operations role, where I spent my time investigating and remedying production issues using my development, investigative and still nascent testing skills. It was a hectic job working long hours away from home. Finding myself overloaded with work, I quickly learned to be a little ruthless with my time when trying to figure out what was broken and what needed to be fixed.

One skill I picked up, was being able to distinguish whether I was researching a bugor trying to find a new bug. When researching, I would be changing one thing or removing something (etc) and seeing if that made the issue better or worse. When looking for bugs, I'd be casting the net wide, trying to catch as many issues as I could.

Nowadays, as a full-time tester, I use both skills every day. When I think I've uncovered a curious result or behaviour I switch to investigating mode to zoom in on prevalence, scope and causes etc. A good programmer will also usually have very good investigative skills which they put into practice when debugging and researching anomalies.

Some people attempt to formalise what we (programmers, testers etc) do, by copying what people appear to be doing and putting that into a script. For example, some teams often take the 'change/try one thing' approach from research & investigation and try to use that -alone- for testing. They use a (BDD, Excel, Notepad etc) script that checks a calculation by calling the feature with example parameters, and checking the answer is correct. Great, Job done, green tick, the feature works, move on. (Ok, maybe not.)

Step 1: How might we improve on that?

That might be useful. But for just a couple more lines of code (basically just a for loop) you could check 100,000 parameters. This scattergun of data is much more likely to catch a bug. Why? Well, for example, there are usually rounding issues, hidden boundaries etc. From my experience there usually are issues, that won't come to light with just a token couple of example musket shots through the logic.

Bess fired a lead bullet 1.9 cm in diameter, but could only fire every 20 seconds.

Step 2: Change of approach

We can use these same techniques and tools when testing in real time. We don’t need to have a fully automated solution straight away. This gives us the ability to also find these issues up front, as we test. This isn’t so much a technology change but rather, a change of mindset.

We can stop thinking of ‘the tests’ as being an immovable, unchangeable detector of ‘good-ness’ (or not bad-ness for things we have checked) and instead think of them as a handy tool that I can grab anytime I need to dig deeper, get a clearer picture of whats going on. The ‘tests’ become something you use, a toolset, rather than something you build.

You will soon find your code is more generalised, more configurable and helps you find bugs - quicker.