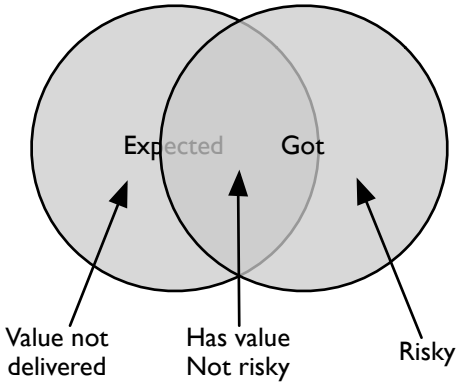



We *need* both exploratory and scripted testing. But this introduces a great chunk of duplicate work, scanning twice through things we not only wanted, but got. What's more, there's no action to be taken - we wanted it, we got it, now move on. What *happens* if you just use one approach?

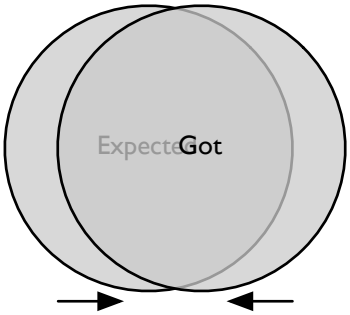
We'll all recognise that if we switch our minds on, sometimes working through our expectations will tell us about things we have received, but didn't expect. Sometimes testing what we've got will tell us about stuff we wanted, but haven't got. Lucky us. Bright, observant us. If your testing is intense enough and your differences small enough, you can rely on just checking what you wanted, or just checking what you've got.

If you're *only* scripting, I expect that your chances of successfully spotting most of the risks will be improved by experience and covert use of exploratory techniques. If you're *only* exploring, I expect that your chances of adequately assessing the value of your new system will be influenced by experience and covert use of scripted test design. I rather expect you're using both, no matter what your strategy says.

o O o

We can push scripted testing further and further upstream, away from the moment of delivery, until we reach a point where our understanding-of-what's-expected is changing too much for us to write scripts economically. But there's more to early scripted testing than getting off the critical path.

When we writing test scripts early, we can put the currently-imaginary system through its paces. We can spot architecture problems before they become part of the design, spot design problems before they become part of the code. Working early can influence the quality of the system, bring what we get closer to what we want. Early scripted testing does this, if it is allowed to, and this potential for overall improvement is one of the abiding strengths of scripted testing. As part of 'quality engineering', scripts can bring what we want and what we get closer together. If they do this, scripts can pay off before exploration even starts.




Late scripted testing cannot improve quality. Script construction that has no effect on the system cannot improve quality. If scripts are made late and in an isolated silo, if they're written to be run once and expensively, their potential is lost - and you're left with a handful of obsolete work-generators that aren't worth the effort, however attractively they may have been engineered, labelled and filed.

o O o

Exploratory testing is stuck on the critical path. It needs to be fast and focussed. Speed and reach will come from tool use. Focus will come from an understanding of the technology, the business, potential for exploitation. Without these, your exploration is hamstrung from the start.

If you start testing, any kind of testing, only at system delivery, it is too late to effectively deal with architectural and design problems. If exploration can be focussed away from user interaction and into deeper matters, you may have an opportunity to go upstream and find unexpected risks in time - but you've still not left the critical path.

One of the biggest slowers-down of exploratory testing are perversely, bugs. Great slews of unattended, anyone-can-find-these, function-masking bugs. Particularly if you're in a situation where you need to re-explore every fix delivery.