The Hunky-dory Hypothesis

10 January 2011 · Scientific Method, testing

“If we just run it with this data, and it looks Ok, we’ll know it works” the architect says expectantly, “Right?”

“You’re right, We might see ‘it work’”, “How would that help?” I answer.

“Well errh, it works, so we can put it live tomorrow.”

We’ve seen this situation before; it can arrive in conversation like the above, or from a review of Acceptance Test results or in a host of other forms. The premise is: everything is fine, we’ve done the work - we have evidence everything is done and working.

But, how can demonstrating that the application can ‘work’ help? How will seeing the acceptance test results as ‘green’ help? That might sound nonsensical, but seriously: how does it help our customer make the decision to ship [or not]? Or help them distribute people and resources better? It may seem that telling them ‘its all hunky-dory’ is news and good news at that, but it isn’t. It can be reassuring, but often it actually harms our understanding of the situation we are in.

Lets take a step back. The code has been pair-programmed, checked by the developers, unit tested and peer reviewed. They believe it’s good, their supervisor/reviewer thinks its great. The acceptance tests are all green and emitting a warm fuzzy feel good glow. The project manager, program manager and technical architect are all confident in their designs and plans - they’re good - the software is ready to ship tomorrow. The vested parties believe and hence in their view, know, its good to go. A few cursory checks (see above ‘test case’) is all that is required. So how will confirming their view help? They’re already convinced, they already know.

Their view is often the default for programmers, system administrators, those without a testing background or those with a vested interest in seeing the system shipped tomorrow. Their mental image of the software is like a clean sheet of paper, its blank, unblemished, all shiny and new. This spotless canvas also reports no information. No shadows, no ambiguities no feedback whatsoever. If during my testing, I confirm the ‘hunky dory’ hypothesis, I’m effectively painting white paint on my canvas.

But if during our testing, we find a problem, an area of ambiguity, or an outright bug - We increase our knowledge of the system under test. By filling in these gaps in our knowledge, We are painting a picture of how our system looks under various conditions. This image, albeit a ‘Negative’, provides the detail that was previously missing from our knowledge. Without this

image we are blind to the potential problems and can not see the risks, let alone be able to mitigate them.

Good testing delivers information that helps us disprove our illusions. This process is invaluable in obtaining accurate knowledge of the risks associated with a software release. In a culture where software is considered innocent, until it crashes or loses you money, good testing can help overcome the group-think that often leads to poor software in live environments.