Was there a test for that? No, and there shouldn't be.

14 March 2018 · agile, automation, critical thinking, experiment, unexpected

The release shipped. For a while, the team felt good. The work was done, the team had achieved something, and that was rewarding.

Unfortunately for the team, it wasn’t long before a problem was found. The Product Owner wasn’t happy and had asked was going on down there in the galley, do we need new coders? Better ones? Hipster coders?

The release shipped…

After an investigation, some blushes, raised eyebrows and a couple of “Oh… Yeeeah’s” they found the cause. A confusion had collided with a bodge, and the result was a mess.

Should they write an automated - test for this problem?

An embarrassing mistake or a misstep can make us feel we have to do something. An action greater than a fix is needed. A penitance needs to be performed, to redeem ourselves, to make us right again.

Sometimes the penitence is best spent adding a test for that issue. Especially if writing that test has a low cost, the frequency of the problem occurring is high or the impact of the problem is substantial.

But often, there is a smarter path. Take for example the opportunity cost (https://www.investopedia.com/terms/o/opportunitycost.asp). While you add that automated test, What else could your team be doing? For example, you might be able to spend the time fixing that underlying bodge.

As a tester, a highly targeted automated test might give an instant feeling of protection and a helpful dose of CYA (https://en.wikipedia.org/wiki/Cover_your_ass). But while you are coding that test - you are not spending as much time:

- Looking for root causes, in code and process
- Searching for similar assumptions the team made
- Thinking laterally about what else might be broken
- Talking to your team about how to stop this happening again.

Also, the automated test may just turn out to be more expensive to write than the fix. Sometimes, iterating is the right thing to do - not just regarding isolating the right product but also in honing its quality.

You can learn from failure, it’s essential you step back & take a moment to let that happen. Investigating Software (http://www.investigatingsoftware.co.uk/) can help you do that.