SHAF
.COM

Test case culture in software testing

By Shafiq Ahmed

Background

As a test engineer, I have been guilty of helping prop up the argument of test cases as being good. It usually stems from the fact that from my experience, most new test engineers have started their careers on a heavy dose of working with test cases, including myself.

Use of test cases has also become a barrier for using test engineers on software projects. Some project managers believe that the only way to test is to use test cases, which can be expensive if the budget and time constraints are tight. It is not true since there are many different methods an experienced test engineer can apply to a project.

Experienced testers don't always follow the test cases to the letter. There are times when a test engineer may deviate from the script based on the use of their own heuristics try to investigate a particular area that might be troubling. You could end up with situations where all the tests are passing, but there are show-stopper issues in the issue tracker.

You could argue that the test engineer did a lousy job of writing the test cases, but that is not the whole story. There is a limited amount of time, and the amount of typing and thinking that goes into writing test cases takes effort, time and budget.

Trying to mechanise the act of testing into easy to follow steps does not always lead to finding important issues in the software. Test cases are useful for showing a particular functionality has been checked; however, they constrain the creative thought process of a test engineer and can be inefficient.

In some organisations, I have seen test cases written by experienced engineers for use by inexperienced test engineers to follow. This approach has the problem of not letting the junior engineer think for themselves, and they can potentially miss important issues.

If you want detailed steps, then you are better off using something along of test charters, which records the steps during test execution and you get more detail and precise steps recorded.

Test cases are not all bad if combined with other testing methods as part of an overall test strategy. The following are based on my experience and are not exhaustive as there are many more ways a test engineer carries out software testing.

Ad-hoc testing

I have used the ad-hoc test technique to raise bugs very quickly to help mature the software before doing any formal testing. It is not "truly ad-hoc" as the test engineer will subconsciously or consciously following some heuristic.

Some managers tend to discourage the use of ad-hoc testing due to lack of visibility. You can measure this by the number of bugs that have been raised or by notes.

This approach can be applied at any stage of a project and with other test methods described below.

Exploratory testing using charters

Using a charter allows test engineers to structure their testing in a way that will aid decision making. A test charter will have details such as description, environment, setup, steps and other relevant information. Recording of the test steps, observations and other issues takes place during the test execution stage.

Using charters does mean adding more time during the test execution stage as there is an overhead in recording the results. In my experience, it takes less time overall as your not spending time upfront on thinking about what the steps might be when the software or function is complete.

This method is also useful for when doing paired testing where one can be the scribe and the other carrying out the testing. Test engineers can swap roles and share their knowledge. You can use additional tools such as screen recorders or a video camera to record your tests.

Afterwards, you can have a debrief with a project manager, software architect or a more experienced test engineer, but this may not be necessary. A debrief can be useful if the software is of a complex nature, there is a high degree of risk, or when using inexperienced test engineers.

I would recommend this approach when testing medical device software, as there is a detailed record of the test execution compared with using test cases test.

I have used this approach in some projects but used it without timeboxing or having a debrief.

Using test titles

I have used this approach in an environment where there is not enough time to write detailed steps for test cases. This method works well if you have access to well-written user requirements/specifications. It also improved the portability of tests across different projects as there is little modification to be made. The titles are open to interpretation, but it does not constrain the test engineer's testing by being too rigid.

Hybrid approach

If you are working on a large project or dealing with complex software, you can combine the different ways of carrying out the testing. Some software features or functions are better off tested using test cases and other functions using exploratory testing.

To contact me visit

www.shaf.com/contact

© Shafiq Ahmed 2019
www.shaf.com