Name: Chun-Chi Huang
Course: Software Testing
Professor: Michael McKee
Date: 06/09/2018

Lesson: Great Planning Accommodates Changes

Recently, high-tech products are rarely being developed without changes. Both the internal voices within the companies and the external requirements from customers usually demand changes while the product is being developed. Consequently, changes are inevitable, and people need to get used to those. In this situation, how testers deal with these difficulties is very important at present. I entirely agree with this idea that testers need to learn how to make a great plan to accommodate changes. This not only helps testers avoid future troubles but also save their time.

First, testers only need to develop tests as they need. One explanation is that testers do not need to generate a large suite of tests in advance. If later changes to this product cannot be applied these tests, at least these tests have been used. The other explanation is that testers need to understand what developers do before testers receive the build. There is no need to create duplicate tests. For example, some developers implement an auto-build mechanism with smoke testing. Once developers deliver the build to testers, this means that this build already passed smoke testing and is qualified for testers to directly perform their regular tests.

Documenting costs time, and testers have to do it properly. Although it is a good habit for testers to write down everything they do, apparently, these extra efforts will occupy too much time. Consequently, testers need to judge whether it is worth recording everything. If documenting the particular test and sharing it with others can benefit the company a lot, testers should do it without hesitation. Otherwise, there is no need to waste time on it.

Based on the concept of reusability, testers should develop the test cases and tools which can be used for cross-platforms and future projects. Nowadays, there are many platforms applied in testing, such as Windows, Linux, etc., and the tools have to support them. Moreover, the user interfaces of projects always change, but the principles are usually similar. If the test cases and tools can be re-used again and again, this will significantly reduce costs for the company.

Smoke testing is a wonderful test to quickly detect basic failures earlier. In my opinion, two types of smoke testing are necessary. The first smoke test is to check whether the new feature, which developers just integrated and want to test, exists in this build or not. If not, there is no reason to proceed with the full-set test. The second smoke test is to detect the fundamental functions in a short time. If those features are functional, this build is eligible for doing future tests. Thus, smoke testing can help disqualify the bad build as soon as possible at a low cost and makes it efficient for testers to deal with frequent builds.

Assisting developers in generating the unit testing can save testers and developers a lot of time. Although creating the unit testing is usually the developers' task, due to their limited schedules, sometimes they are not willing to do it. Testers are encouraged to negotiate with the developer manager and politely ask developers to implement the unit testing. In general, the unit testing is integrated into the daily-build system and performed for every build. This will efficiently detect the failures before developers generate a formal build.