Name: Chun-Chi Huang
Course: Software Testing
Professor: Michael McKee
Date: 04/26/2018

Why Is Unit Testing Hard?

"Why is unit testing hard?" is a complicated question, and the answer should be "It depends." If the developer creates a class which only has one method, addition, it should not be a difficult task to perform the unit testing. The developer only provides two input values which include the boundary values and then checks the result to see whether or not these calculated values match the expectation. Then this task will be finished. However, in fact, unit testing is not usually that simple. When developers are required to implement unit testing, they are usually struggling with external dependencies. Sometimes, they are also unable to determine what exactly to test and how to exactly run the test. The other challenge is dealing with legacy code bases because some of these code bases are untestable.

Handling external dependencies should be a big challenge for developers. They are not able to directly implement test cases by only depending on the particular logics, because these logics use the information that comes from external dependencies. Developers cannot ignore them since these external dependencies are necessary. Consequently, developers need to create mocks, stubs or anything else to match the logics in the target unit. After that, the implementation of this unit testing can be meaningful. In many cases, these external dependencies have their own special states, and developers have to consider carefully and implement cautiously. It is painful for developers to go through all of those steps.

The second challenge is the difficulty in determining what exactly to test and how exactly to test it. For performing the unit testing, developers will depend on the method level to generate the test cases. However, in some cases, there is a lot of implementation and logic existing in one method. Consequently, developers need to find a way to separate them and produce several test cases for only one method. As for how to test, this also involves the first challenge, external dependencies. What kind of external dependencies developers want to create? How deep external dependencies should be involved? These questions are always in the back of the developers' mind.

In addition, facing legacy code bases can be a nightmare for developers. Some of these legacy code bases have lasted for more than one decade. No one can guarantee these code bases were developed according to the concept of being testable. In this situation, developers will likely struggle with either implementing unmeaningful test cases or refactoring this legacy code base. Both approaches are not developers' first desire. Creating unmeaningful unit testing will not lead to satisfying results, but refactoring the whole legacy code base will cost too much. That is why the legacy code base is a significant challenge for unit testing.