Ideal vs. Non-Ideal Testing Pyramids

William T. Thomason

WEB-430

Assignment 3.3

Ideal vs. Non-Ideal Testing Pyramids

Automated testing is a vital step in developing and deploying an application, in order to accomplish this a gentleman by the name of Mike Cohn thought of the “ideal testing pyramid” concept. By using the ideal testing pyramid, you spend more time creating more unit tests that can find the errors faster, earlier and cheaper. The non-ideal testing pyramid, the inverse is happening by spending more time manual testing and less on unit testing.

First let’s talk a little about the importance of automated testing. Building, testing and deploying an ever-increasing amount of software manually soon becomes impossible — unless you want to spend all your time with manual, repetitive work instead of delivering working software. Automating everything — from build to tests, deployment and infrastructure — is your only way forward. In the beginning software testing was overly manual work done by deploying your application to a test environment and then performing some black-box style testing by clicking through the application as a user trying to find bugs in your application. It becomes apparent very quickly that doing this all manually can be extremely time consuming and difficult to keep the test environment the same. By automating this process, you can keep manipulate the test environment any way you want, and the testing all runs automatically and consecutively freeing your time for other tasks.

Anyone serious about automated testing their software should use the Ideal testing pyramid. The test pyramid consists of three layers, at the bottom you have unit tests, then service tests and at the top you have user interface tests. The higher you go on the ideal pyramid (UI test) the more it costs and the slower the testing is accomplished. The lower you are on the pyramid (unit tests) the less it costs and the faster the tests can get accomplished. Let’s talk about the layers of the pyramid a little more.

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. Unit tests directly interact with product code, meaning they are “white box.” Typically, they exercise functions, methods, and classes. Unit tests should be short, sweet, and focused on one thing/variation. They should not have any external dependencies – mocks/monkey-patching should be used instead.

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and teststubs are used to assist in IntegrationTesting. Integration tests cover the point where two different things meet. They should be “black box” in that they interact with live instances of the product under test, not code. Service call tests (REST, SOAP, etc.) are examples of integration tests.

End-to-end testing is a technique used to test whether the flow of an application right from start to finish is behaving as expected. The purpose of performing end-to-end testing is to identify system dependencies and to ensure that the data integrity is maintained between various system components and systems. End-to-end tests cover a path through a system. They could arguably be defined as a multi-step integration test, and they should also be “black box.” Typically, they interact with the product like a real user. Web UI tests are examples of integration tests because they need the full stack beneath them.

Automated testing is key to developing a reliable high-quality application. The methods and principles used to accomplish this task are crucial. The ideal testing pyramid in a must have in your corner when testing your application.

References

Vocke, H. (2018, February 26). The Practical Test Pyramid. Retrieved July 28, 2019, from <https://martinfowler.com/articles/practical-test-pyramid.html>

Cochran, T. (2018, May 28). Test Pyramid: the key to good automated test strategy. Retrieved July 28, 2019, from <https://medium.com/@timothy.cochran/test-pyramid-the-key-to-good-automated-test-strategy-9f3d7e3c02d5>

Automation Panda. (2018, July 31). The Testing Pyramid. Retrieved July 28, 2019, from <https://automationpanda.com/2018/08/01/the-testing-pyramid/>