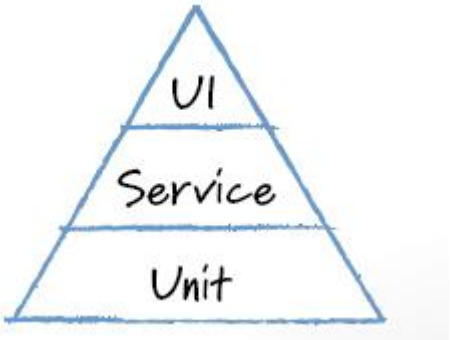
The Forgotten Layer of the Test Automation Pyramid

by Mike Cohn • 125 Comments

The Forgotten Layer of the Test Automation Pyramid

Even before the ascendancy of agile methodologies like Scrum, we knew we should automate our tests. But we didn't. Automated tests were considered expensive to write and were often written months, or in some cases years, after a feature had been programmed. One reason teams found it difficult to write tests sooner was because they were automating at the wrong level. An effective test automation strategy calls for automating tests at three different levels, as shown in the figure below, which depicts the test automation pyramid.



At the base of the test automation pyramid is unit testing. Unit testing should be the foundation of a solid test automation strategy and as such represents the largest part of the pyramid. Automated unit tests are wonderful because they give specific data to a programmer—there is a bug and it's on line 47. Programmers have learned that the bug may really be on line 51 or 42, but it's much nicer to have an automated unit test narrow it down than it is to have a tester say, "There's a bug in how you're retrieving member records from the database," which might represent 1,000 or more lines of code. Also, because unit tests are usually written in the same language as the system, programmers are often most comfortable writing them. Let's skip for a moment the middle of the test automation pyramid and jump right to the top: the user interface level.

Automated user interface testing is placed at the top of the test automation pyramid because we want to do as little of it as possible. Suppose we wish to test a very simple calculator that allows a user to enter two integers, click either a multiply or divide button, and then see the result of that operation. To test this through the user interface, we would script a series of tests to drive the user interface, type the appropriate values into the fields, press the multiply or divide button, and then compare expected and actual values. Testing in this manner would certainly work but would be brittle, expensive, and time consuming. Additionally, testing an application this way is partially redundant—think about how many times a suite of tests like this will test the user interface. Each test case will invoke the code that connects the multiply or divide button to the code in the guts of the application that does the math. Each test case will also test the code that displays results. And so on. Testing through the user interface like this is expensive and should be minimized. Although there are many test cases that need to be invoked, not all need to be run through the user interface. And this is where the service layer of the test automation pyramid comes in. Although I refer to the middle layer of the test automation pyramid as the service layer, I am not restricting us to using only a service-oriented architecture.

All applications are made up of various services. In the way I'm using it, a service is something the application does in response to some input or set of inputs. Our example calculator involves two services: multiply and divide. Service-level testing is about testing the services of an application separately from its user interface. So instead of running a dozen or so multiplication test cases through the calculator's user interface, we instead perform those tests at the service level. Where many organizations have gone wrong in their test automation efforts over the years

has been in ignoring this whole middle layer of service testing. Although automated unit testing is wonderful, it can cover only so much of an application's testing needs. Without service-level testing to fill the gap between unit and user interface testing, all other testing ends up being performed through the user interface, resulting in tests that are expensive to run, expensive to write, and brittle.

For more on Scrum and agile testing, pick up a copy of *Succeeding with Agile*.

Posted: December 17, 2009

Tagged: programming, testing
