Test doubles in modern software development

Mocks, Spys, Fakes and Stubs.
What they are and how to use them.

First Databank

By Stuart Hopwood

Test Doubles

A Test Double is a generic term for any case where you replace a production object for testing purposes.

- Martin Fowler

Test Doubles

Test Doubles can be created with a test framework such as Moq or NSubstitute, or handwritten.

Most of the time a test framework is quicker and easier to use. However there are times when a handwritten test double is a better fit.

Test Double Vocabulary

What should I call these things?

Test Double Vocabulary

Test Doubles can come with a few names, each with their own intended use case.

They are...

DUMMY

Used to fill out a method’s parameters.

Not commonly used.

STUB

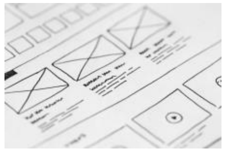
Returns a call to the method with a pre-programmed output.

Require setting up with every test.

FAKE
NEWS

FAKE

Same as a Stub but handwritten.



MOCK

Set up with expectations of the calls they are to receive.

Used to verify that a method call has been triggered correctly or at all.

SPY

A handwritten Mock.

Does not drink Martini.

Command / Query Separation

The two categories of methods

Command / Query Separation

There are essentially only two categories of methods...

Command methods

And Query methods.

Command / Query Separation

A good practice is to divide an object's methods into those two separated categories.

This practice was named: Command Query separation by Bertrand Meyer in his book "Object Oriented Software Construction".

COMMAND

Modifies the state.

Does not return the state.

QUERY

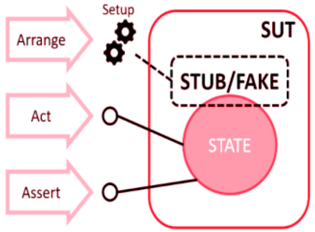
Does not modify the state.

Fetches and returns the state.

What Test Double Should I Use?

When should you use a Stub or Fake?

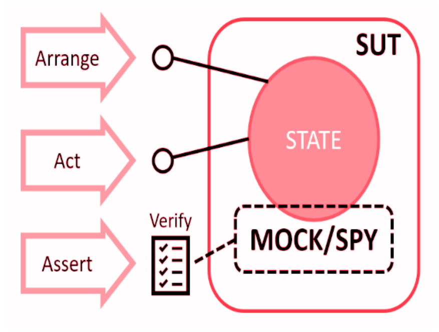
Use Stubs for Queries



A query is a method that fetches and returns state data. It should not modify that data.

Stubs respond to the method call with a predetermined value.

Use Mocks for Commands



A command is a method that modifies the state of application.

Mocks verify that the method call was triggered and even how many times it was called.

Should I test Interfaces?

Yes! Always

Testing Interfaces

Tests should verify public behaviour.

We test the interface to test the *behaviour*, not a specific concrete implementation.

Should the implementation change, our tests don’t need to change.

Some Guidelines

Test Double Guidelines

Don’t add behaviour inside Test Doubles

Don’t use Test Doubles for isolated objects

Don’t create too many Test Doubles

Any Questions?

Thank you for listening!

Sources

Alcor Acadamy

Martin Fowler https://martinfowler.com/bliki/TestDouble.html

Command Query separation by Bertrand Meyer https://www.amazon.com/Object-Oriented-Software-Construction-Book-CD-ROM/dp/0136291554

Pragmatists Blog https://blog.pragmatists.com/test-doubles-fakes-mocks-and-stubs-1a7491dfa3da

The end.