Effective C# Unit Testing for Enterprise Applications

LEARNING THE BASICS OF EFFECTIVE UNIT TESTS

Rusty Divine
SOFTWARE CONSULTANT
@CornerPosts cornerpostsoftware.com

Core Concepts

Definitions used in this course

What are effective unit tests

Review of the code used in this course

Effective Unit Tests

Prerequisites of Effective Unit Tests

Clean code

Testable design

Context-aware

Understanding of unit testing

Qualities of Effective Unit Tests

Clear and simple

High value

Flexible

Definitions

Unit of Work

Everything that happens from invoking a public method to it returning the results after it's finished; it’s the work done along the path you see the debugger take through your code.

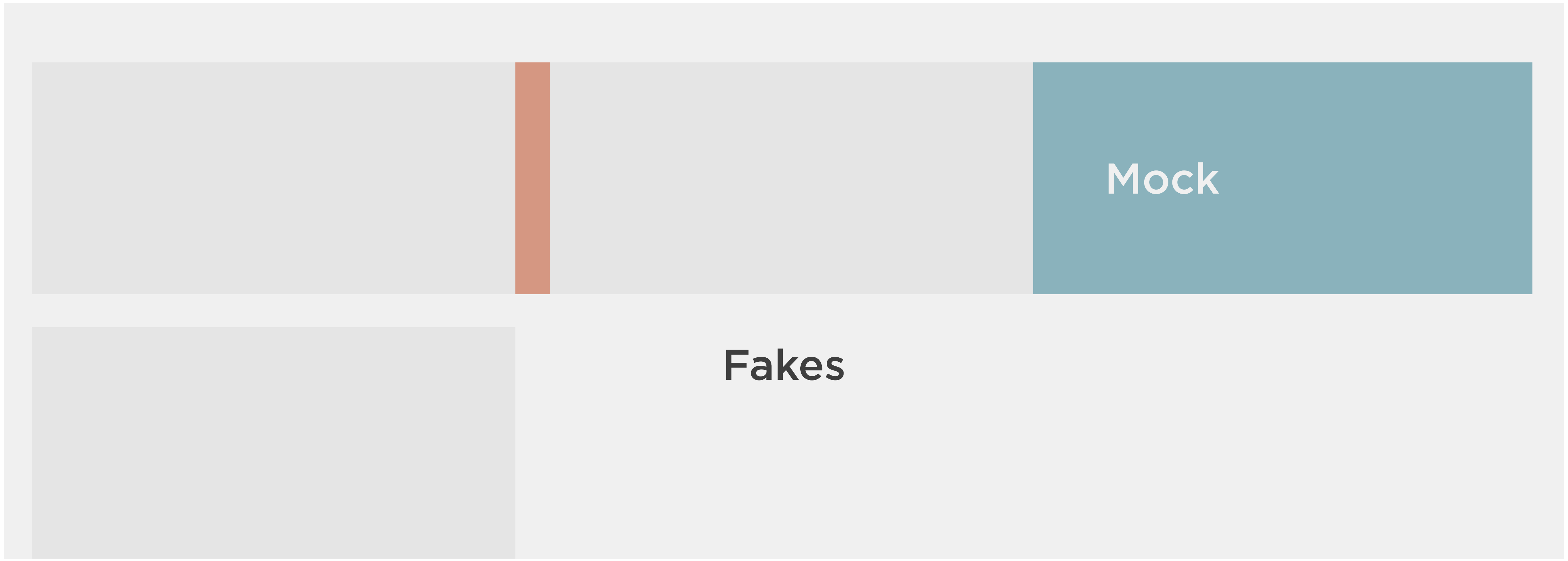
Unit Test

Code that invokes a unit of work within the confines of a project layer while faking external dependencies and validates an assumption about one specific scenario.

Integration Test

Code that invokes a unit of work that crosses project boundaries, uses actual external dependencies, and/or validates many different aspects about the code under test.

Stubs, Mocks, and Fakes



Stub

A substitute for a dependency in the code under test that allows the code to compile and the dependency to return data as specified by the test but importantly cannot itself directly make a test fail.

Mock

A substitute for a dependency in the code under test that knows how many times each of its methods were called an in what order so that it can validate an assumption about how the dependency was used and therefore make a test fail.

Fake

A generic term for a replacement of a real dependency with something the test specifies, which includes both stubs and mocks.

Demo

Overview of the solution

Summary

Unit tests inform design

Unit of work, unit test, integration test

Stubs, mocks, fakes

Clarity, simplicity, value