Mocking with Moq and NUnit

GETTING STARTED WITH MOQ

Jason Roberts
.NET DEVELOPER

@robertsjason dontcodetired.com

Overview

Demo code overview

Write a test without mocks

Install the Moq mocking library

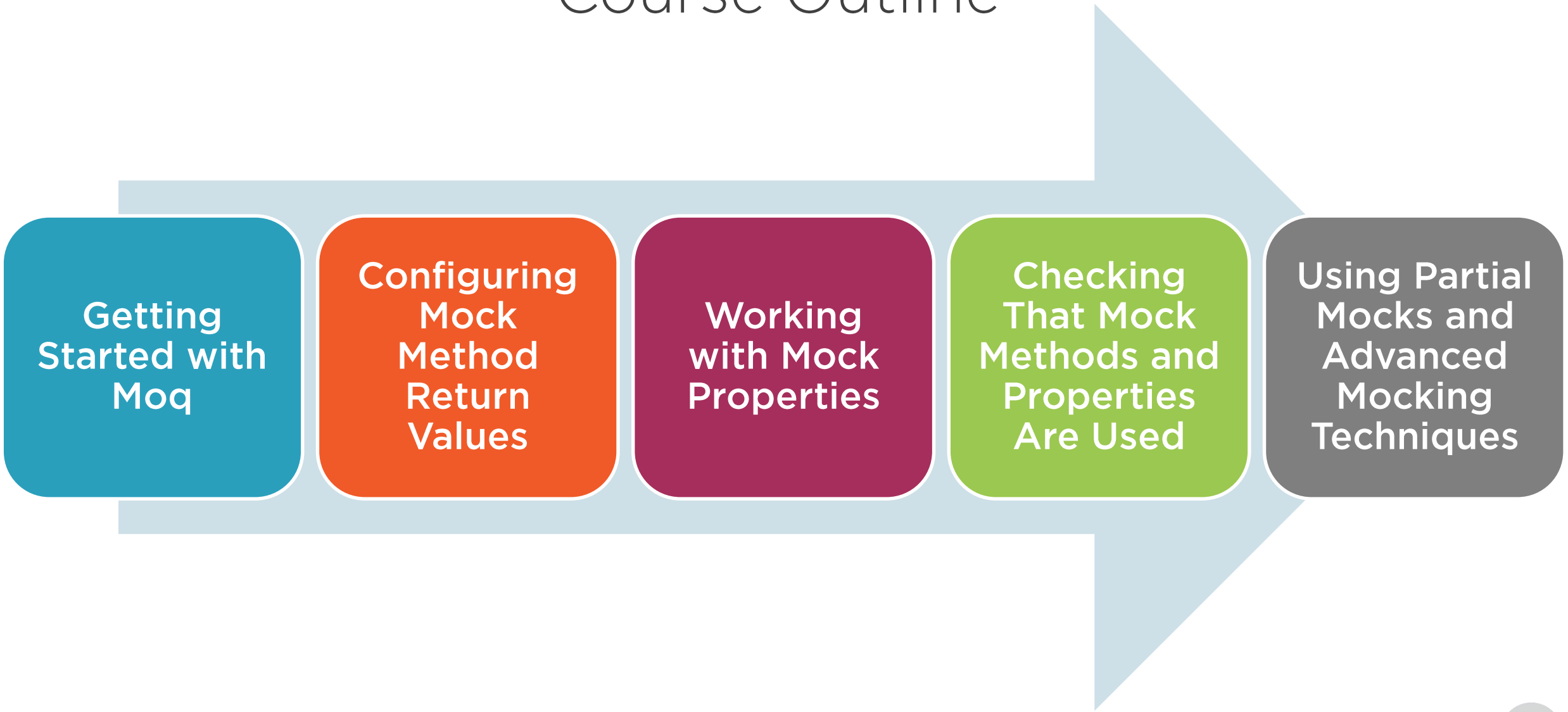
Create an mock object using Moq

An overview of mocking

- Why mock?
- What is a unit?
- Fakes, dummies, stubs, and mocks

Overview of the Moq library

Course Outline

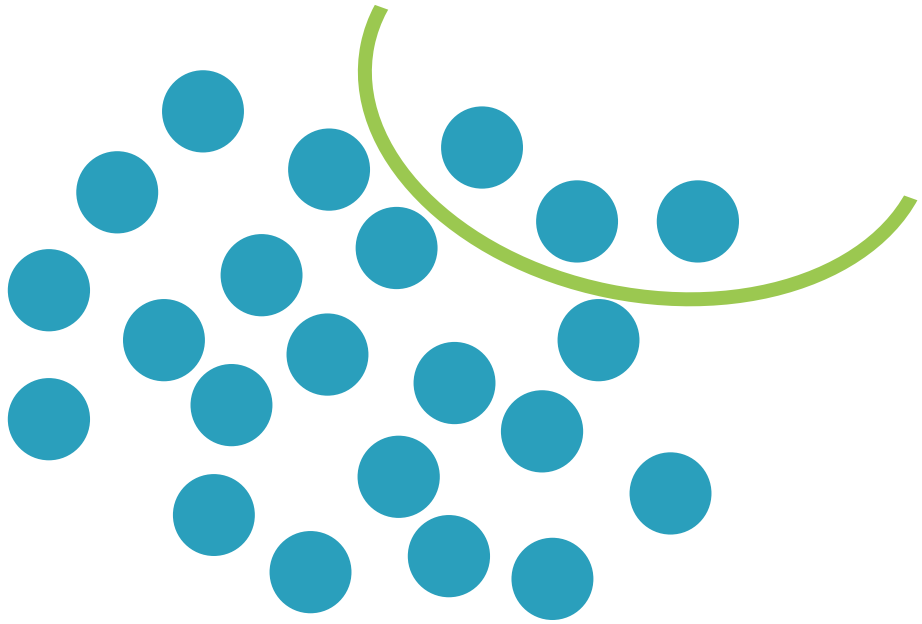


Suggested prerequisite course:

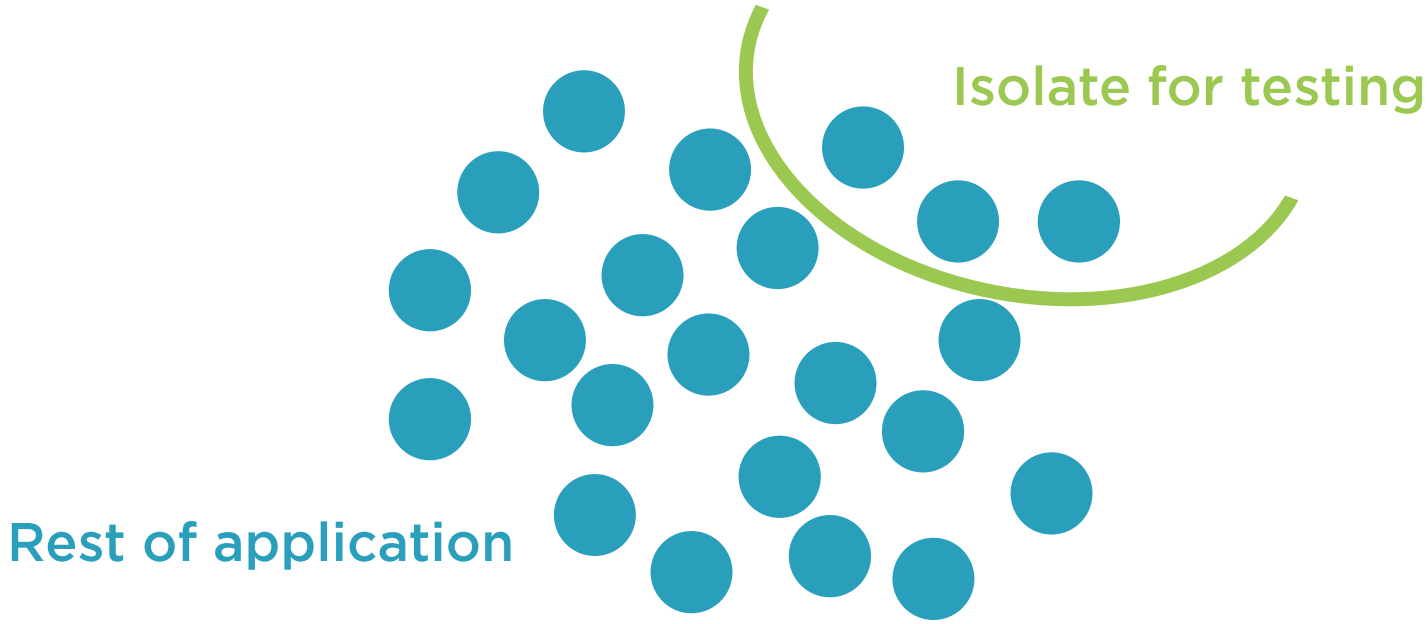
Introduction to .NET Testing with NUnit 3

http://bit.ly/psnunit3

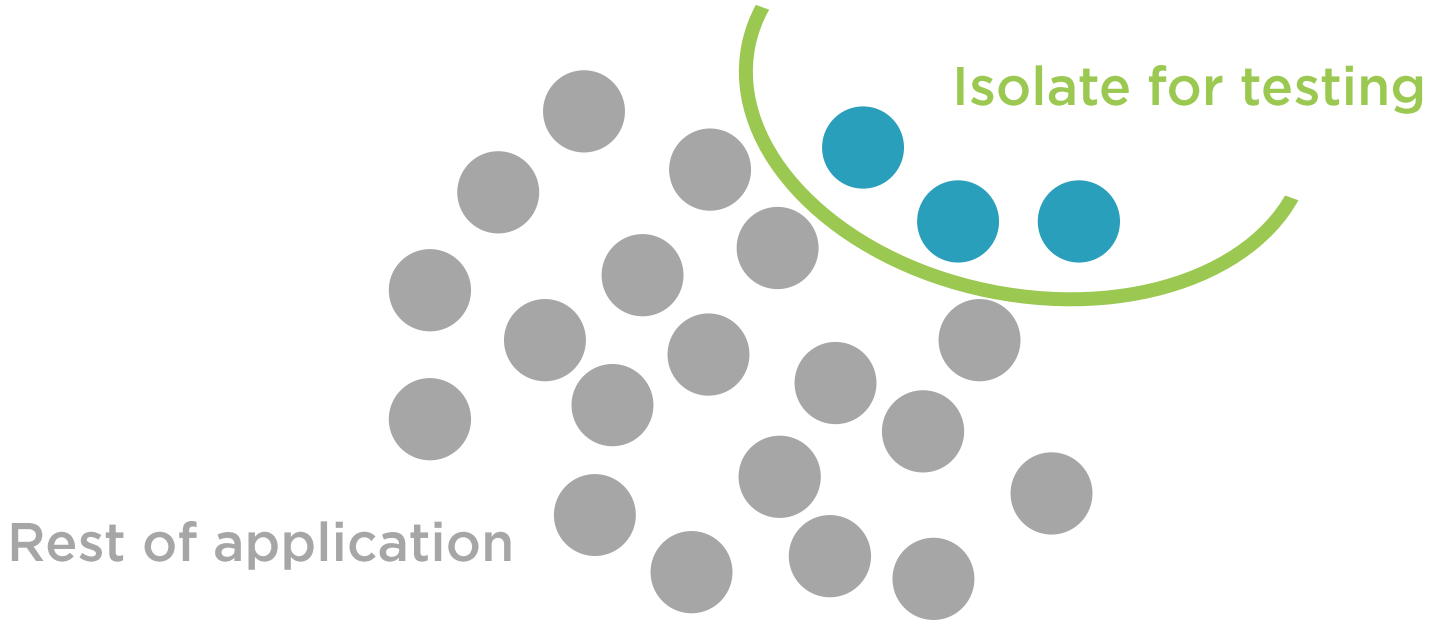
An Overview of Mocking



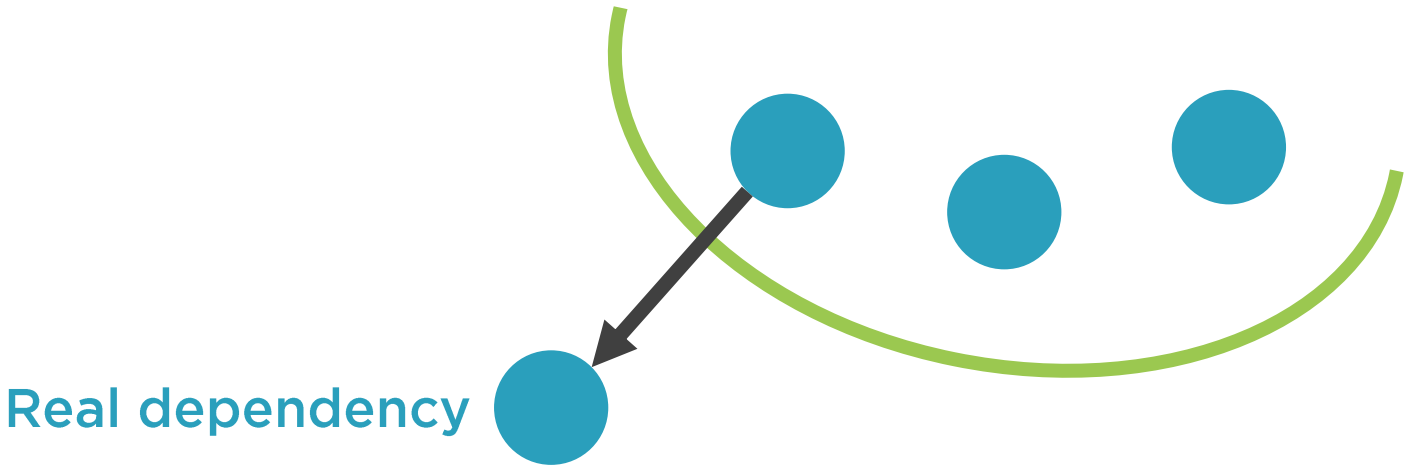
An Overview of Mocking



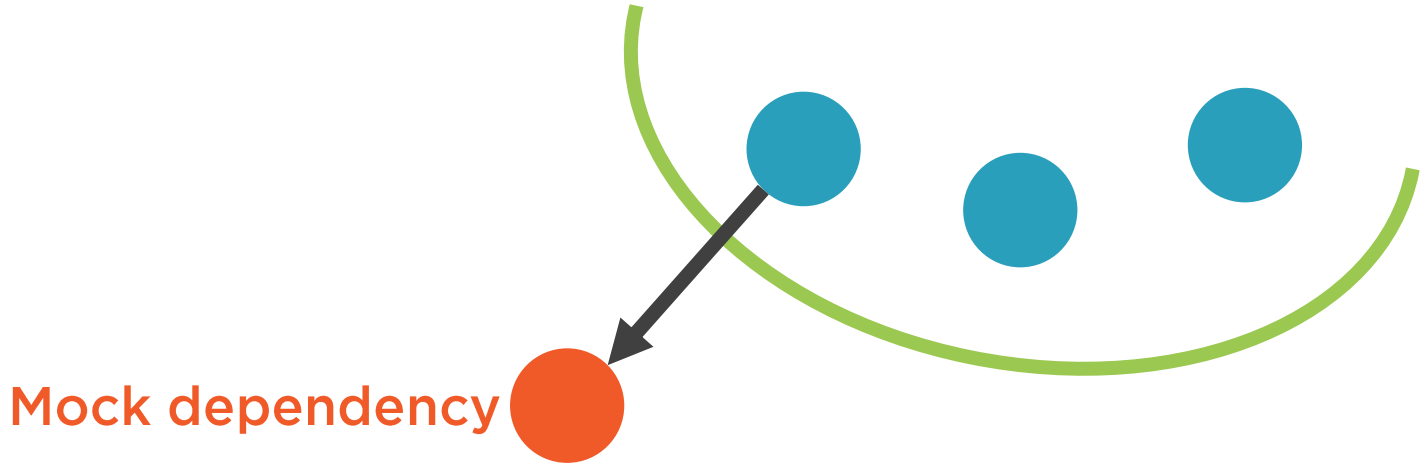
An Overview of Mocking



An Overview of Mocking



An Overview of Mocking



Mocking

Replacing the actual dependency that would be used at production time, with a test-time-only version that enables easier isolation of the code we want to test.

Why Mock?

Reduced test complexity

Improved test execution speed

- Slow algorithms & external resources

Non-deterministic dependencies

- E.g. DateTime.Now

Support parallel development streams

- Real dependency not yet available
- Another team, external contractor, etc.

Improve test predictability/reliability

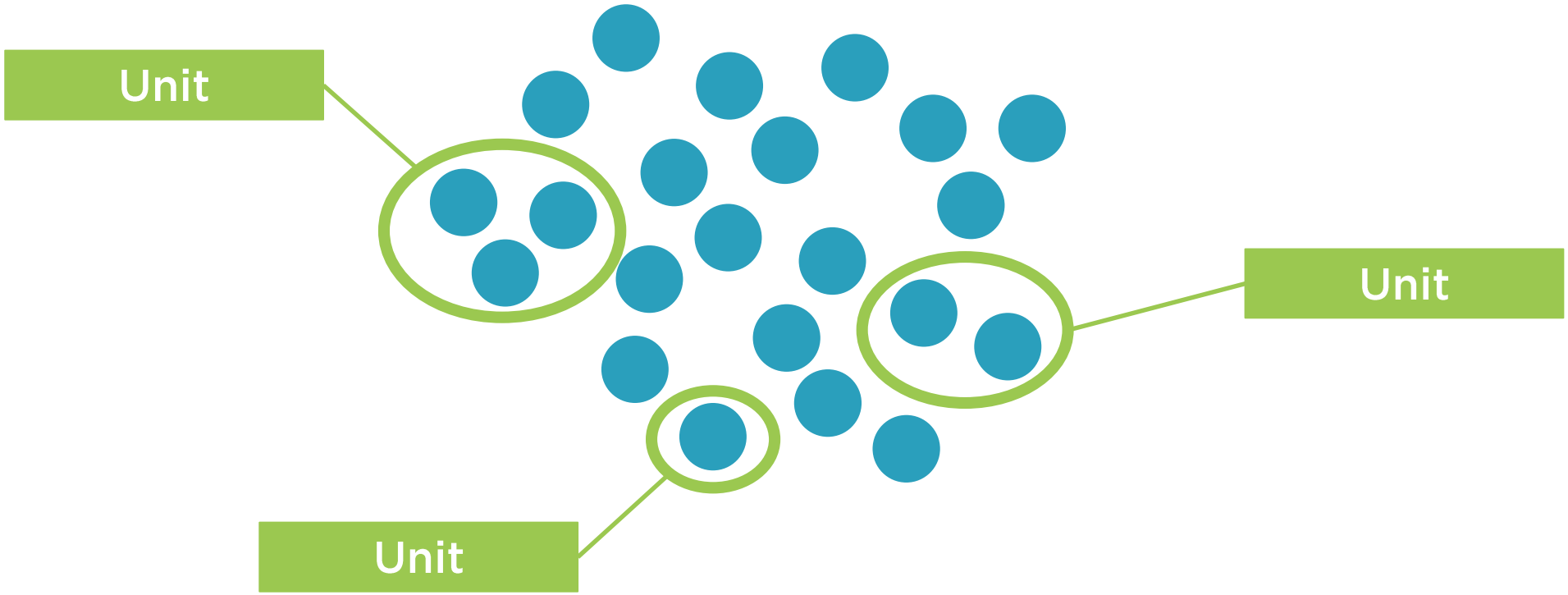
Reduce testing costs

- External dependency cost per usage
- Interfacing with internal mainframe

In addition to testing things in isolation, things should also be tested working together with real dependencies.

(Integration, API, UI tests, etc.)

Unit Tests



“...it's a situational thing - the team decides what makes sense to be a unit for the purposes of their understanding of the system and its testing”

Martin Fowler

https://martinfowler.com/bliki/UnitTest.html

There are many terms used to try and differentiate between different types of “mock” objects.

Fakes, Dummies, Stubs, and Mocks

Fakes

Working implementation

Not suitable for production

EF Core in-memory provider

Dummies

Passed around

Never used

Satisfy parameters

Stubs

Provide answers to calls

Property gets

Method returns

Mocks

Verify interaction

Properties

Methods

Moq

Fakes, Dummies, Stubs, and Mocks

Test Double

Fakes	Dummies	Stubs	Mocks
Working implementation	Passed around	Provide answers to calls	Verify interaction
Not suitable for production	Never used	Property gets	Properties
EF Core in-memory provider	Satisfy parameters	Method returns	Methods

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

Martin Fowler

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

In this course, for simplicity, we'll use the term "mock object".

An Overview of Moq

Design goals:

- Simple
- Practical
- Easy to use
- Strongly-typed

Pronounced “Mock-you” or “Mock”

Open source

https://github.com/moq

44,683,882 total NuGet downloads

Summary

Demo code overview

Wrote a test without mocks

- Null dependencies

Installed Moq from NuGet

Created initial mock objects using Moq

- new Mock<IIdentityVerifier>()
- new Mock<ICreditScorer>()

An overview of mocking

Why mock?

What is a unit?

Fakes, dummies, stubs, and mocks

Overview of the Moq library

Up Next:

Configuring Mock Method Return Values