WHO LET THE MOCKS OUT?

…A look at Testing using Test Doubles

Matt Webb – 16/05/2023

Introduction

Testing

Essential to the development process

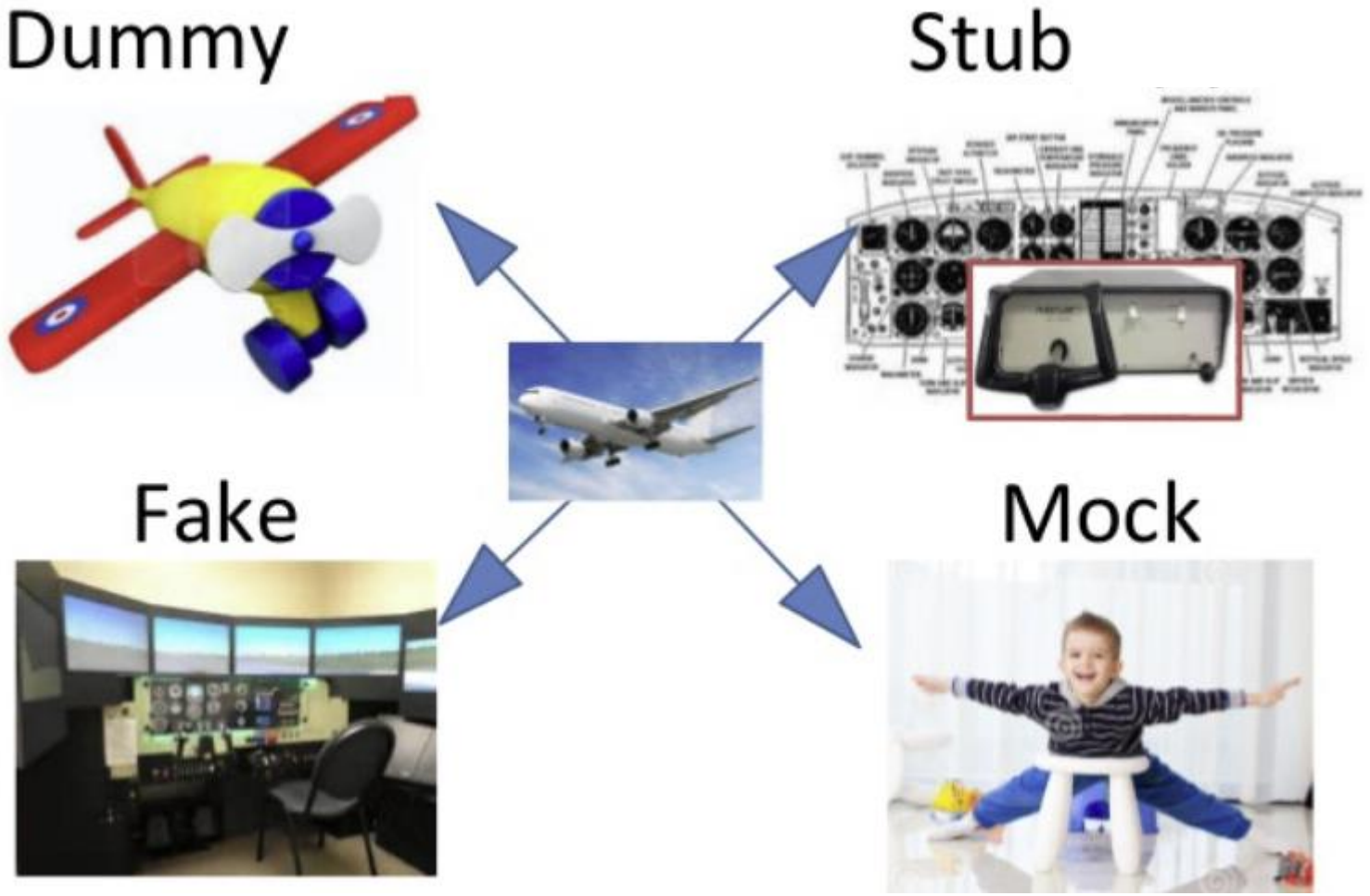
Ensures desired requirements are met with robust code

Several techniques used to ensure desired behaviour

TEST DOUBLES

What Are Test Doubles?

- Dummies
- Stubs
- Fakes
- Mocks



What is a Dummy?

- ✓ Object passed to a method to satisfy parameter list
- ✓ Does not contain any logic
- ✓ Does not affect behaviour of the method
- ✓ Often passed as parameter, but not used in methods logic

Dummy Example

```
1 reference
public class OrderProcessor
{
    private readonly ILogger _logger;

    0 references
    public OrderProcessor(ILogger logger)
    {
        _logger = logger;
    }

    0 references
    public void ProcessOrder(Order order)
    {
        // Process the order logic here...

        // Log the order processing information
        _logger.Log( message: $"Order processed: {order.OrderId}");
    }
}
```

```
2 references
public interface ILogger
{
    1 reference
    void Log(string message);
}
```

Production Code

```
2 references
public class DummyLogger : ILogger
{
    1 reference
    public void Log(string message)
    {
        // Do nothing in the dummy logger
    }
}
```

```
0 references
public class OrderProcessorTests
{
    [TestMethod]
    0 references
    public void ProcessOrder_Should_Log_Order_Processed()
    {
        // Arrange
        var dummyLogger = new DummyLogger();
        var orderProcessor = new OrderProcessor(dummyLogger);
        var order = new Order { OrderId = 123 };

        // Act
        orderProcessor.ProcessOrder(order);

        // Assert
        // We can't assert anything specific about the logging in this case,
        // since the dummy logger doesn't do anything. But if there were other
        // aspects of the order processing we could assert them here.
    }
}
```

Test Code

What is a Stub?

- ✓ Object that returns a predefined response
- ✓ Control indirect inputs by simulating specific conditions or return values
- ✓ Simpler than the real dependencies, focusing on behaviour
- ✓ Simulate the behaviour of an unavailable object

Stub Example

```
2 references
public class ProductService
{
    private readonly IProductRepository _productRepository;

    1 reference
    public ProductService(IProductRepository productRepository)
    {
        _productRepository = productRepository;
    }

    1 reference
    public bool IsProductInStock(int productId)
    {
        var product = _productRepository.GetProductById(productId);
        return product != null && product.StockQuantity > 0;
    }
}

2 references
public interface IProductRepository
{
    1 reference
    Product GetProductById(int productId);
}

1 reference
public class Product
{
    0 references
    public int ProductId { get; set; }
    0 references
    public string Name { get; set; }
    1 reference
    public int StockQuantity { get; set; }
}
```

Production Code

```
1 reference
public class StubProductRepository : IProductRepository
{
    2 references
    public Product GetProductById(int productId)
    {
        // Create a stub product with the desired properties
        if (productId == 1)
        {
            return new Product
            {
                ProductId = 1,
                Name = "Test Product",
                StockQuantity = 5
            };
        }
        else
        {
            return null; // Return null for any other product IDs
        }
    }
}

0 references
public class ProductServiceTests
{
    [TestMethod]
    0 references
    public void IsProductInStock_Should_Return_True_If_StockQuantity_Greater_Than_Zero()
    {
        // Arrange
        var stubProductRepository = new StubProductRepository();
        var productService = new ProductService(stubProductRepository);
        var productId = 1;

        // Act
        var result = productService.IsProductInStock(productId);

        // Assert
        Assert.IsTrue(result);
    }
}
```

Test Code

What is a Fake?

- ✓ **Simulates a real object with simplified functionality**
- ✓ **Typically used when real object is difficult to use or unavailable for testing.**
- ✓ **Unlike a stub, it may have some simplified functionality, compared to the real object**
- ✓ **Often used in scenarios where complexity or dependencies are not desired during testing.**

Fake Example

```
1 reference
public class EmailService
{
    1 reference
    public virtual bool SendEmail(string recipient, string subject, string body)
    {
        // Implementation of sending an email
        // This could involve network operations, SMTP configuration, etc.
        // For testing purposes, we don't want to send actual emails.
        // So, we create a fake implementation that logs the email details instead.
        Console.WriteLine($"Fake email sent to: {recipient}, Subject: {subject}, Body: {body}");
        return true;
    }
}

0 references
public class EmailServiceFake : EmailService
{
    1 reference
    public override bool SendEmail(string recipient, string subject, string body)
    {
        Console.WriteLine($"Fake email sent to: {recipient}, Subject: {subject}, Body: {body}");
        return true;
    }
}
```

Production Code

```
[TestClass]
0 references
public class EmailServiceTests
{
    [TestMethod]
    0 references
    public void SendEmail_Should_Log_Email_Details()
    {
        // Arrange
        var emailService = new EmailServiceFake();
        var recipient = "test@example.com";
        var subject = "Test Subject";
        var body = "Test Body";

        // Act
        var result :bool = emailService.SendEmail(recipient, subject, body);

        // Assert
        Assert.IsTrue(result);
        // Additional assertions can be made to check the log or output as needed.
    }
}
```

Test Code

…And Finally….Mocks

- ✓ Programmed to imitate the behaviour of real objects
- ✓ Provides the ability to test interactions between objects
- ✓ Verifies that methods were called and how
- ✓ Set expectations on method calls and return values

Mock Example

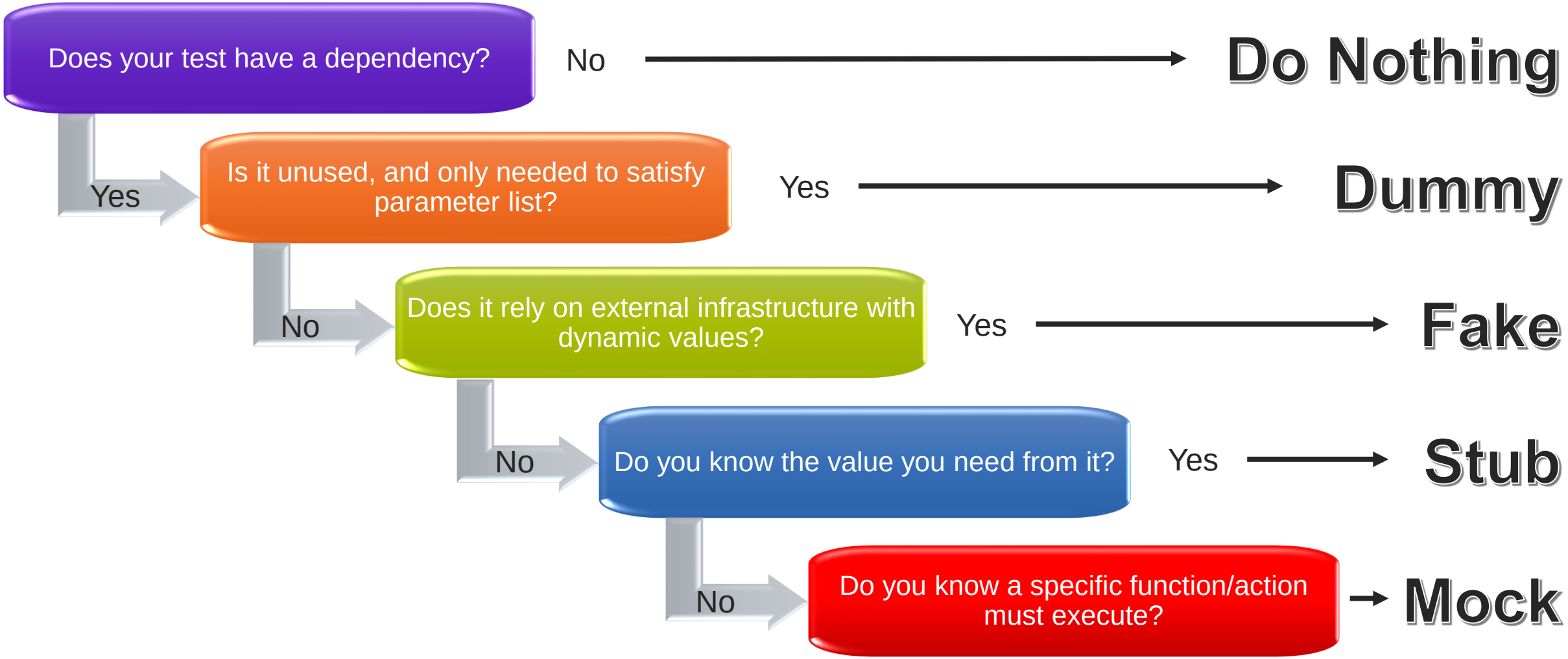
```
2 references
public interface IWeatherService
{
    1 reference
    string GetWeather(string location);
}

1 reference
public class WeatherReporter
{
    private readonly IWeatherService _weatherService;

    0 references
    public WeatherReporter(IWeatherService weatherService)
    {
        _weatherService = weatherService;
    }

    0 references
    public string GetWeatherReport(string location)
    {
        var weather :string = _weatherService.GetWeather(location);
        // Generate a weather report based on the weather data
        return $"The weather in {location} is {weather}.";
    }
}
```

Production Code

```
[TestClass]
0 references
public class WeatherReporterTests
{
    [Test]
    0 references
    public void GetWeatherReport_Should_Return_Weather_Report()
    {
        // Arrange
        var weatherService = Substitute.For<IWeatherService>();
        var reporter = new WeatherReporter(weatherService);
        var location = "New York";
        var expectedWeather = "Sunny";

        weatherService.GetWeather(location).Returns(expectedWeather);

        // Act
        var result :string = reporter.GetWeatherReport(location);

        // Assert
        Assert.AreEqual( expected: $"The weather in {location} is " +
                                   $"{expectedWeather}.", actual: result);

        weatherService.Received(1).GetWeather(location);
    }
}
```

Test Code

Decision Tree – What Test Double To Use



Thank you for Listening…..

Questions?